Prediction, interpolation and extrapolation of drilling data with Deep Learning

by

Andrzej Tunkiel

Thesis submitted in fulfilment of the requirements for the degree of
PHILOSOPHIAE DOCTOR (PhD)

Supervisors

Dan Sui

Tomasz Wiktorski

Faculty of Science and Technology
Department of Energy and Petroleum Engineering

2022

University of Stavanger
N-4036 Stavanger
NORWAY
www.uis.no

ISBN 978-82-8439-134-2
ISSN 1890-1387

PhD Thesis UiS no. 675

"The first principle is that you must not fool yourself and you are the easiest person to fool."

Richard P. Feynman

Preface

This dissertation is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy (PhD) at the Department of Energy and Petroleum Engineering at the University of Stavanger, Norway. The PhD program was supervised by Prof. Dan Sui and co-supervised by Prof. Tomasz Wiktorski in the period of August 2019 to January 2022.

This dissertation is written based on five scientific papers, presented in Chapter 5, and reproduced in the Appendix.

Abstract

Directional drilling is an established technology within the petroleum industry. In this traditionally conservative and risk averse environment application of artificial intelligence encounters difficulties at various stages of the process. Additionally, the unique nature of drilling data makes deployment of the off-the-shelf algorithms problematic on multiple levels.

An artificial neural network with a custom architecture is explored in this dissertation; it is combining recurrent elements as well as traditional artificial neurons to fully utilize information in both the dynamic behaviour of the system, as well as data that is available in real time. This fit for purpose algorithm unlocks significant improvements to the traditional directional drilling technologies.

To complement this novel architecture, research is presented that focuses on the data preparation. Methods are presented that tackle characteristic problems of the real-time drilling logs that make the incompatible data digestible for the machine learning. New algorithms were developed that allow to gauge the difference between the raw and the processed data.

Simplifying the field deployment a method for on-the-job training is explored for the developed architecture, where no prior knowledge about the exact drilling system nor historical data is required. To aid the confidence in the presented methods a fit for purpose sensitivity analysis is investigated allowing to peek inside the data driven algorithm.

Acknowledgements

I would like to thank Prof. Dan Sui and Prof. Tomasz Wiktorski for all the help extended to me throughout the course of my studies.

I would like also to thank Øystein Arild for creating an atmosphere of support and thrust in the Department of Energy and Petroleum Engineering.

Lastly I would like to thank my wife Kasia for supporting me throughout the whole PhD journey.

List of publications

1. Tunkiel, A. T., Sui, D., & Wiktorski, T. (2021). **Impact of data preprocessing techniques on recurrent neural network performance in context of real-time drilling logs in an automated prediction framework** - Journal of Petroleum Science and Engineering 2021[AT1]

2. Tunkiel, A. T., Sui, D., & Wiktorski, T. (2021). **Reference dataset for rate of penetration benchmarking**. Journal of Petroleum Science and Engineering.[AT2]

3. Tunkiel, A. T., Wiktorski, T., & Sui, D. (2020). **Continuous drilling sensor data reconstruction and prediction via recurrent neural networks**. In ASME 2020 39th International Conference on Ocean, Offshore and Arctic Engineering. American Society of Mechanical Engineers Digital Collection. [AT3]

4. Tunkiel, A. T., Sui, D., & Wiktorski, T. (2021). **Training-while-drilling approach to inclination prediction in directional drilling utilizing recurrent neural networks**. Journal of Petroleum Science and Engineering. [AT4]

5. Tunkiel, A. T., Sui, D., & Wiktorski, T. (2020). **Data-driven sensitivity analysis of complex machine learning models: A case study of directional drilling**. Journal of Petroleum Science and Engineering.[AT5]

Contents

Preface iii

Abstract iv

Acknowledgements v

List of publications vi

Abbreviations x

Symbols xiii

1 Introduction 1
- 1.1 Objectives . . . 2
- 1.2 Methodology . . . 3
- 1.3 Novelty . . . 4
- 1.4 Scientific results . . . 5
- 1.5 Engineering applications . . . 6
- 1.6 Dissertation outline . . . 7

2 Data preparation methods 9
- 2.1 Source of the data . . . 9
  - 2.1.1 Data logging . . . 10
- 2.2 Data quality improvements . . . 11
  - 2.2.1 Data imputation . . . 12
  - 2.2.2 Data resampling methods . . . 19
- 2.3 Benchmarking datasets . . . 25

3 Machine learning modelling procedure 28

3.1 Introduction to Machine Learning . . . 28
3.2 Data split in data series . . . 29
3.2.1 Feature engineering . . . 31
3.2.2 Attribute selection . . . 33
3.3 Recurrent Neural Networks . . . 36
3.3.1 RNN cell structure . . . 38
3.3.2 Long Short-Term Memory . . . 39
3.3.3 Gated Recurrent Unit . . . 40
3.4 Training-while-drilling approach . . . 41
3.5 Case study of directional drilling . . . 43
3.5.1 Branched model architecture . . . 45

4 Sensitivity analysis **49**
4.1 Rationale . . . 49
4.2 Definition . . . 50
4.2.1 Sensitivity analysis in Machine Learning . . . 51
4.2.2 Sensitivity analysis of ML in drilling . . . 52
4.3 Local, global, and data-driven sensitivity . . . 52
4.3.1 Partial Derivative . . . 53
4.4 Data driven sensitivity . . . 54
4.4.1 Sensitivity per input channel . . . 55
4.5 Comparison with Sobol' indicies . . . 57

5 Contributions **59**
5.1 Publications . . . 59
5.1.1 Publication AT1 . . . 60
5.1.2 Publication AT2 . . . 61
5.1.3 Publication AT3 . . . 62
5.1.4 Publication AT4 . . . 63
5.1.5 Publication AT5 . . . 64
5.2 Petroleum perspective, machine learning perspective . . . 65
5.3 Other published research . . . 66
5.3.1 Volve dataset exploration . . . 66
5.3.2 Automated gap filling . . . 67
5.3.3 Drilling data quality improvement . . . 67
5.3.4 Downhole data correction . . . 68

6 Conclusions **69**
6.1 Future work . . . 71

A Appendix, publications **91**

B Impact of data pre-processing techniques on recurrent neural network performance in context of real-time drilling logs in an automated prediction framework, AT1 **92**

C Reference Dataset for Rate of Penetration Benchmarking, AT2 **159**

D Continuous Drilling Sensor Data Reconstruction and Prediction via Recurrent Neural Networks, AT3 **196**

E Training-while-drilling approach to inclination prediction in directional drilling utilizing recurrent neural networks, AT4 **222**

F Data-driven sensitivity analysis of complex machine learning models: A case study of directional drilling, AT5 **260**

Abbreviations

AI	Artificial Intelligence
AMISS	AutoMated Imputation for SerieS
BHA	Bottom Hole Assembly
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRISP-DM	Cross-industry standard process for data mining
CSV	Comma Separated Values
CUDA	Compute Unified Device Architecture
DDSI	Data Driven Sensitivity Index
DLS	Dog-leg Severity
DNI	Directional and Inclination (sensor)
DRV	Data Reconciliation and Validation
FAST	Fourier amplitude sensitivity testing
FRN	Fixed Radius Neighbour
GC	Gap Coefficient
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HQHP	High Quantity High Percentage (gaps)

HQLP	High Quantity Low Percentage (gaps)
KNN	K-Nearest Neighbours
LQHP	Low Quantity High Percentage (gaps)
LQLP	Low Quantity Low Percentage (gaps)
LSTM	Long Short-Term Memory
LWD	Logging While Drilling
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MD	Measured Depth
MNIST	Modified National Institute of Standards and Technology (database)
MSE	Mean Squared Error
MSE_g	Riemann sum squared of function $g(\cdot)$
ML	Machine Learning
MLP	Multi Layer Perceptron
MWD	Measurement While Drilling
NLP	Natural Language Processing
OAT	One At a Time
OMAE	International Conference on Ocean, Offshore & Arctic Engineering
PaD	Partial Derivative
PCA	Principal Component Analysis
RAM	Random Access Memory
RFE	Recursive Feature Elimination
RMS	Root Mean Square
RMSD	Root-mean-square deviation
RMRS	Root Mean Riemann Squared
RNN	Recurrent Neural Network
RNR	Radius Neighbour Regressor
RPM	Revolutions Per Minute

ROP	Rate of Penetration
RSS	Rotary Steerable System
S_g	Riemann sum of function $g(\cdot)$
SS_g	Riemann sum squared of function $g(\cdot)$
SVM	Support Vector Machine
TL	Total Length
USROP	University of Stavanger Rate Of Petentration (dataset)
VARS	Variogram Analysis of Response Surfaces
WITSML	Wellsite Information Transfer Specification Markup Language
WMAPE	Weighted Mean Absolute Percentage Error
WOB	Weight On Bit
XGB	eXtreme Gradient Boosting

Symbols

symbol	name	unit
δ_{cf}	cutoff value (for Gap Coefficient)	
$\theta_{th}\%$	threshold	%
R_T	error in Rolls theorem	
y_t	true value (in prediction)	
$\hat{y}_t$	predicted value (in prediction	
φ	activtion function	
σ	standard deviation, sigmoid function	
h_t	RNN, output	
v_t	RNN, recurrent connection value	
W	RNN, input weight	
U	RNN, previous output weight	
$tanh$	RNN, hypebolic tangent function	
f_t	LSTM, activation vector of forget gate	
b_t	LSTM, bias	
i_t	LSTM, activation vector of input gate	
o_t	LSTM, activation vector of the output gate	
$\tilde{c}_t$	LSTM, activation vector of cell input	
c_t	LSTM, cell state vector	

z_t	GRU, update gate vector
r_t	GRU, reset gate vector
$\hat{h}_t$	GRU, candidate activation vector
ΔS	sensitivity, small change to the input
h	RNN output

Chapter 1

Introduction

Artificial intelligence (AI) is a collection of approaches and algorithms that experiences both explosive growth in research and industry. It is a technology that promises self driving cars, automatic translations, and replacing humans in various jobs and positions. While AI is a term that became widely known to the public, the practical side of it, machine learning, is what is being worked on by scientists and engineers alike.

Machine learning (ML) [1], has a clear definition. In general, it is an algorithm that based on experience in a form of input-output pairs, typically referred to as training data, can predict the outputs of new and unknown inputs. Most basic example of that is fitting a linear regression model which nowadays can be done in Excel with a mouse click. On the other hand, complex algorithms such as reinforcement learning, are able to perform complex tasks like playing computer games [2].

There are countless ML algorithms in existence with new ones being constantly invented. Their architecture is tailored for various types of problems, from simple mapping of arbitrary data to image analysis, translation, or data generation. When focus is set to drilling data and drilling logs, one of the most promising

architectures is the family of recurrent neural networks (RNN) [3]. This mathematical construct can capture temporal information about the signal, building on not only on the information about the current state, but also on a number of the past states. This technology is widely used commercially in translation, voice recognition, music composition and speech synthesis, but has seen only modest adoption in the typically conservative petroleum industry.

Research presented in this dissertation is focused on utilizing the recurrent neural networks for data extrapolation, interpolation and prediction, as well as on data processing and model evaluation. It focuses on the best way of utilizing this technology, finding its potential, limitations, understanding better how does it work, as well as how to correctly apply it in the field.

1.1 Objectives

The general objective of the presented research is to attack some of the unsolved petroleum related problems with the latest machine learning technologies. It is about finding ways of applying modern algorithms to practical real life problems that are bound by constraints unique to the field of oil and gas.

At the same time it is crucial to stress what is not the objective of the presented research. It is not about creating new machine learning algorithms for a broad use. It is an interdisciplinary work, probably best summarized as applied artificial intelligence. It is about a holistic approach to the task at hand. There are four core areas of the presented research:

- **Data preparation.** RNNs have specific requirements when it comes to data, and drilling logs have a very specific structure. Making them compatible was to a large extent not explored in detail before
- **Model creation.** Modern ML algorithms can often be connected together as if they were LEGO bricks, to create new, fit for purpose architectures.

Drilling processes require innovative ML models to efficiently model the downhole environment

- **Model application.** Exploring how a model can be deployed is a critical aspect of applied machine learning. Investigation into how to practically apply the model can define its usability.
- **Model evaluation.** Machine Learning models are typically considered so called *black boxes*, indicating that it is for all intents and purposes not clear how they work. One of the goals of this research is to shed light on what can be expected of the model.

1.2 Methodology

Research presented in this dissertation is following the *Cross-industry standard process for data mining* (CRISP-DM) [4]. This is the most common methodology used for broadly understood data science projects. It consists of the following major phases:

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modeling
5. Evaluation
6. Deployment

All these phases, with the exception of deployment, were executed in the presented work. The *business understanding* is linked to the needs of the industry; this part is linked to the case study presented in section 3.5 as well as in [AT3, AT4], which discusses the problem of delayed sensor readings in relation to directional

drilling using the bent sub assembly. This is a practical and common struggle in the development of directional drilling tools, where it is desired to have the direction sensor as close to the bit as possible. There is a strong business need to move the sensor forward, be via mechanical means or through machine learning.

Data understanding that covers data collection, data description, discovery and data quality evaluation, was performed in publication [AT6], which is not explicitly included in this dissertation. Data utilized in the presented research comes from the Volve dataset [5].

Data preparation step is the sole focus of the most recently published publication [AT1], where various novel pre-processing techniques are applied with focused on solving common drilling data series issues. Since one of the goals of the undertaken research was the application of recurrent neural networks in drilling, the publication focuses on data preparation in light of this technology.

Modelling and *evaluation* phases of the CRISP-DM process dominate publications [AT3, AT4]. RNN based ANN of a custom branched architecture is developed and is evaluated based on a scenario where training is done during drilling. Additional publication [AT5] focuses purely on evaluation of the developed models from the sensitivity analysis point of view.

1.3 Novelty

Top 3 most important novel elements of the presented research are listed below:

- Data pre-processing as described in [AT1] provides a number of novel tools and methods that are aimed at data series preparation with RNN utilization in mind.
- Custom ANN branched architecture utilizing RNN and MLP elements for problems where significant amount of information is carried both in the temporal information of the data series, as well as in parallel data-channels.

This architecture plays a major and minor roles in most of the publications linked to this dissertation [AT1, AT3, AT4, AT5].

- Sensitivity Analysis of ML models utilizing a dataset as a reference starting point for PaD analysis [AT5] is particularly novel and important in relation to RNN-based models that can contain hundreds of inputs, and require sensitivity analysis not only per individual input, but also as a series of inputs.

1.4 Scientific results

Within publication [AT1] one of the explored problems is infilling missing data that stems from varying sensor rates; it was discovered that there is a statistically significant difference between forward filling and linear interpolation. The same publication highlights the efficiency of using PCA decomposition instead of traditional attribute selection for drilling data, as well as proposes a mathematical method of evaluating how close resampled data reflects the original dataset with erratic recording frequency.

In [AT3] an RNN-based ANN with custom architecture significantly outperforms traditional ML approaches. A case study of sensor lag in directional drilling is explored, where the inclination sensor is present in the MWD over 23 meters behind the bit in the BHA utilizing bent sub assembly. The same model is later utilized in [AT4] where it proves the efficiency of continuous learning approach when deploying an untrained model in the field, with all the training performed *on the job*. A high resolution analysis is performed to evaluate how deep one has to drill and train the model utilizing freshly harvested data to start yielding satisfying results.

Publication [AT5] presents a novel approach to sensitivity analysis, and provides effective tools for understanding the relationships between the inputs and the outputs of the ML model. It is an approach rooted in one-at-a-time sensitivity analysis method that offsets some of the limitations of such methods by performing

the analysis with the train/test dataset as a collection of starting points, and presents the results as a statistical distribution for additional clarity.

The work [AT2] focuses on providing a catalyst for future research in a form of reference dataset for ROP modelling. It contributes also a stark example how strongly the data split into training and testing data influences the accuracy of various models.

1.5 Engineering applications

There are a number of potential engineering applications that could utilize methods developed and presented throughout this dissertation. One of the key elements is the novel ANN architecture that combines the RNN and MLP elements, allowing for utilization of the most relevant data. The case study of the delayed sensor data in case of directional drilling with bent sub motor was the catalyst for this research and in itself is a valuable practical application. There are a number of potential alternative applications to this architecture:

- predicting directional data in directional drilling ahead of the sensor, allowing for more precise well placement in legacy drilling systems
- drilling tool design optimization through deprioritization of the directional sensor location
- predicting drilling parameters, used in the drilling optimization tasks
- modelling of novel, unconventional drilling systems, where physics are not yet fully understood
- non-petroleum problems, such as household power usage usage prediction, battery degradation prediction, snow cover estimation.

Developed pre-processing techniques as well as sensitivity analysis tools are in principle domain-agnostic, and therefore are potentially applicable to other ML

problems utilizing RNN or other architectures based on taking into account multiple steps in the data-series, such as transformers.

1.6 Dissertation outline

This is a publication based dissertation, where the key research is already published in either a journal or as a peer reviewed conference publication. This document connects all the published research into a cohesive work.

The presented research is interdisciplinary, at the connection between petroleum and data science, what is probably best summarized as applied ML.

Chapter 1 acts as an introduction and sets the stage for the problem.

Chapters 2, 3, and 4 focus on background information and theory, only highlighting the direction of the performed research. They are presented to contextualize the attached publications. Chapter 2 itself focuses on the data itself. It discusses the actual dataset used in this research, established pre-processing methods particularly useful for RNN deployed in drilling; it also puts emphasis on the new research performed in relation to gap classification, gap filling, resampling, and evaluation of the resampling performance. Lastly, work on establishing better access to the datasets to the community is showcased.

Chapter 3 focuses on the core of the machine learning - the model and its application. Proposed custom branched neural network is discussed in detail; it was developed for the purpose of tackling various problems related to real-time drilling.

Chapter 4 presents a new approach to sensitivity analysis, rooted in partial derivative, where sensitivity is explored utilizing the original dataset that provides the starting points of the analysis. This attempts to mitigate the key problem of one-at-a-time sensitivity analysis methods, where selection of the starting point can influence the sensitivity results greatly.

Chapter 5 provides an overview of published research and links them directly to the areas of this dissertation, explicitly highlighting the innovation areas and the main contributions to science. These five key publications have fixed reference keys [AT1, AT2, AT3, AT4, AT5][1] with further citations of other work of my authorship and co-authorship are numbered per their appearance in the text.

Chapter 6 provides summary and conclusions.

Appendix contains relevant published publications that were reproduced in a re-formatted to be consistent with this dissertation. References in the appendix have independent numbering and independently listed bibliographies. No changes to text nor figures were done and the contents are considered for all intents and purposes identical with the published version.

[1] References are generally numerical, with additional AT prefix (f.ex. [AT4]) indicating publications that were personally authored, or co-author.

Chapter 2

Data preparation methods

2.1 Source of the data

The backbone of presented research is the Volve dataset published by Equinor [6]. Volve field was discovered in 1998 with production starting in February 2008. It is located in the central part of the North Sea, nearly perfectly between Stavanger and Aberdeen, the Norwegian and British respective *oil capitals*, where the sea depth is approximately 80 meters. The reservoir itself lies 2700 - 3100 meters deep.

Throughout the life of the field 4.7 billion Norwegian kroner were invested into the Volve field yielding production of 63 million barrels of oil. Production ceased in 2016 with the facilities shut down in 2018.

In June 2018 Equinor has announced that to support innovation and research in the oil and gas industry complete data of that field will be shared. This was originally done using Creative Commons CC BY-NC-SA 4.0 license that was superseded by the Equinor Open Data License in 2020. Both licenses are very open allowing researchers to work on the dataset free of charge, publish results and modify the dataset as long as the original license is retained.

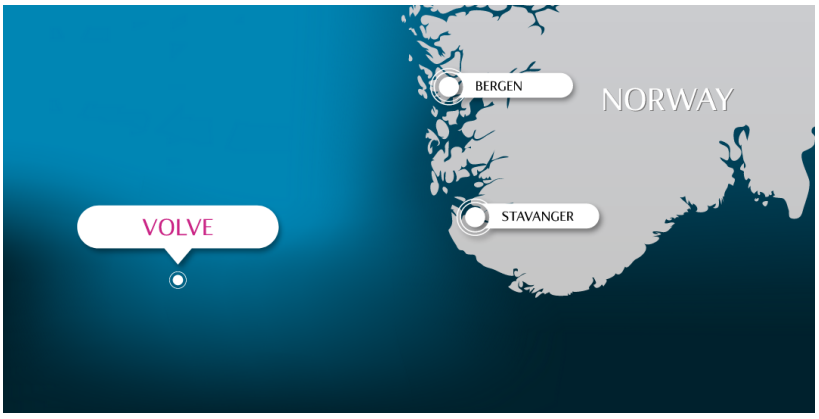


Figure 2.1: Location of the Volve field [6]

The unprecedented amount of data that was made available enabled hundreds of research articles to be written and published. Without a doubt in the absence of the publication of the Volve field data, research presented in this dissertation would not exist.

Within the total 40 000 files shared the most relevant data for research presented in this dissertatiin were the WITSML real-time drilling data. This 5 GB archive contains logs that are both time-based or depth-based spanning multiple wells. Some of the logs were merely few hundred samples long, with the biggest ones containing millions of rows.

2.1.1 Data logging

There are a number of logged attributes connected to the process of drilling. Some can be measured directly, such as standpipe pressure or RPM, with a simple sensor at the surface. Some attributes are measured indirectly, such as WOB, where apparent weight of the drill string is measured, but additional effects must be taken into account such as buoyancy, friction etc. to estimate the actual weight pressed on the drill bit.

There are also attributes measured downhole by systems called Measurement While Drilling (MWD) and Logging While Drilling (LWD). The main difference

between the two is that MWD is typically related to the drilling process itself, while LWD is considered mostly with geology. It is the MWD tool that will measure the inclination and azimuth of the BHA; this information will be transmitted to the surface while drilling to the directional driller, so he or she may place the well according to the well plan.

Despite the progress in communication technologies the most common way of data transmission between the drill bit location and the surface is mud pulse telemetry. Special *flaps* are extended from the MWD tool into the annulus, and by varying obstruction to the return mud flow a digital signal can be created. This signal is later decoded at the surface and the measurements can be read. The biggest limitation of this system is bitrate, typically up to 40 bits per second depending on exact technology used, which is the same as a skilled Morse code operator. As there are multiple sensors in the MWD and LWD the complete transmission sequence of all desired parameters takes typically a couple of minutes.

Differences in sensor polling rates, data transmission limitations etc. create data logs that are not very clean, full of data gaps and that require a lot of processing effort to be used by the modern tools of data science.

2.2 Data quality improvements

Drilling data itself has often multiple quality issues. Recent publication [AT7] lists following problems that are most commonly encountered:

- invalid data - from outside of the range of the measurement system
- inaccurate data - noise and outliers
- incomplete / missing data - various reasons for lack of data
- inconsistent data - result of varying polling frequencies
- redundant data - two sensors recording in principle the same phenomenon or parameter

Similar issues were also encountered in a laboratory scale rig [7] with research focusing on noise filtering. Recent literature also discusses data reconciliation and validation (DRV) in context of wired drillpipe [8], which reduces the error by filtering noises and making the data conform to the mathematical models whenever possible. Overall various schemes are proposed for improving data quality [9] indicating the overall important of this issue.

2.2.1 Data imputation

Drilling logs can be generally considered as a two-dimensional matrix built of rows and columns. Rows designate consecutive readings, while columns are assigned to specific sensors or otherwise calculated attributes. While the order of rows generally matters, the order of the columns does not. Among the attributes in a given dataset there is typically one that can be considered an index. This will be time in the time-based data and measured depth in the depth based data.

This matrix in the case of Volve data is not filled in completely. Some cells are be empty, which is not to be confused with a value of zero. There are also cells that will carry a so-called *sentinel value*, typically designated as an highly improbable reading such as -999.98. Presumably logging software responsible for given sensors is unable to deal with empty values, hence indicates a lack of value through such sentinel number. Throughout the volve real-time drilling data gaps typically occupy over 80% of the cells [AT6].

Gap size identification and classification

There are a number of reasons for gaps to exist in a dataset, that can be generally put into four different categories based on their nominal quantity and the percentage of the dataset they occupy [AT7] with Figure 2.2 visualizing the division. This dissertation focuses on gaps from the upper portion of this chart; gap classification is discussed in more detail, both relating to source of the gaps as well as potential of filling them in the related publication [AT7].

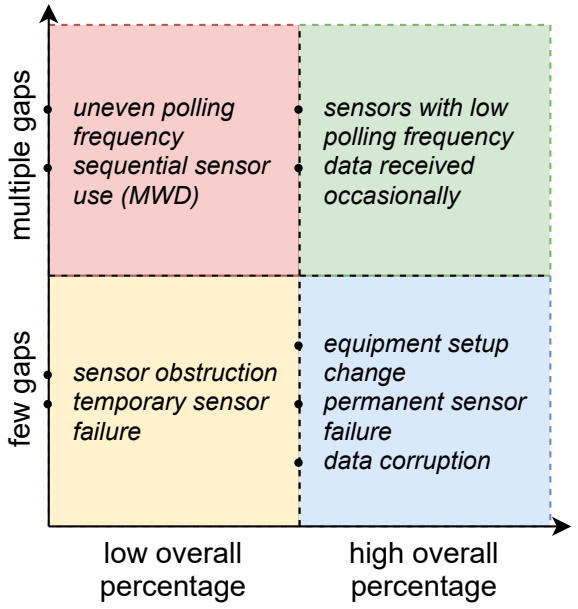


Figure 2.2: Classification of gap sizes

Gap statistics can be visualized as a bar chart, such as the one presented in Figure 2.3, where gaps statistics of Weight on Bit attribute are analyzed for the well F9a of the Volve dataset. Cells that contain data cover 31% of the dataset, with most of the data missing due to gaps that are between one and six rows long. There is also a group of gaps, 177 and 322 rows long, that did not contribute significantly to the data loss percentage wise. However, the length of those gaps is concerning and indicate that they are likely due to data actually missing.

There is a definite spectrum; at one end, there are small and few gaps, that are of least concern. On the other end there are long gaps that are a definite data loss. To quantify that spectrum an equation is proposed to calculate a gap coefficient, GC. This method was published and applied in practice [AT1].

$$GC_i = \frac{i}{GQ_i \cdot TL}. \tag{2.1}$$

Here i is the gap length, GC_i is the gap coefficient for a gap length of i, GQ_i designates the quantity of gaps of length i and TL is the total length of the log.

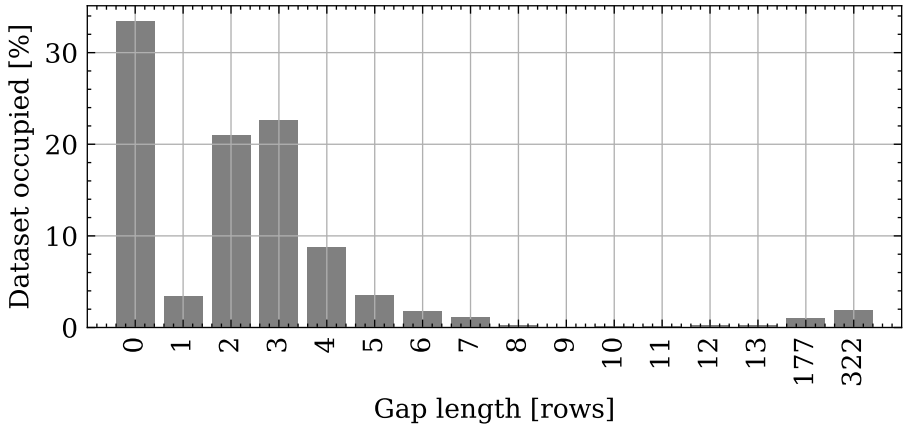


Figure 2.3: Raw gap statistics, Weight on Bit [kkgf], Volve well F9a.

Note that unit here is row count. This equation makes GC proportional to the relative gap size, and inversely proportional to the quantity of the gaps.

Having calculated the GC we can plot the results again, as it is shown in Figure 2.4. The long gaps, quite clearly, have a significantly higher GC. Having manually explored the dataset and reviewed some of the attributes it is possible to establish a cutoff value, δ_{cf}, that will separate data gaps *fillable* with basic methods, such as forward filling or linear interpolation, from the ones being a clear sign of data loss. For the real-time drilling data in the Volve dataset a cutoff value of $\delta_{cf} = 0.005$ was selected, which reliably differentiated between two types of gaps; this cutoff value is also shown in Figure 2.4. This value was selected based on manual inspection of a number of logs in the Vovle dataset. Calculating GC generates a scalar that can be used to identify problematic gaps and was proven useful in automating data processing for RNN studies [AT1].

Literature review

Topic of data imputation is very broad. Different methods will be relevant for data series, when there is significant correlation between consecutive samples [10, 11], and other will be appropriate where no such correlation exists in the tabulated data [1].

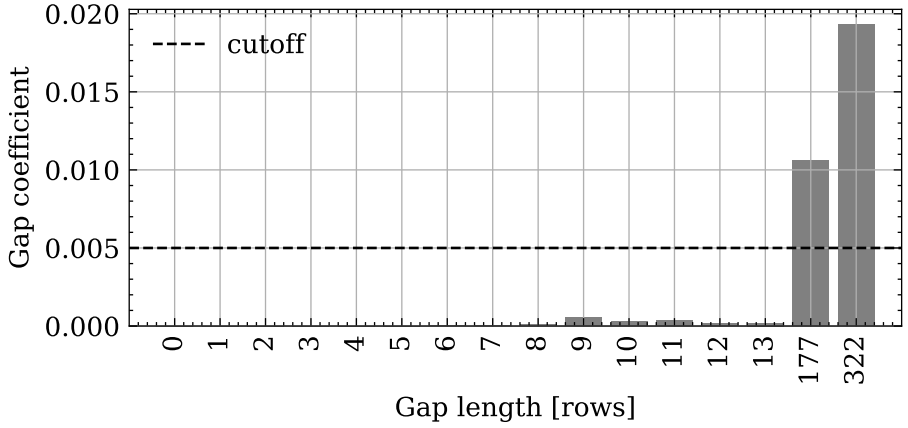


Figure 2.4: Gap statistics, gap coefficient, Weight on Bit [kkgf], Volve well F9a.

This chapter focuses on data imputation where missing values are caused by uneven logging frequency of the measurements systems, that is when various sensors are providing data at different times, and therefore not all attributes will have updated values for all the data rows. This process is therefore more akin to signal upsampling or image upscaling than traditional imputation.

Upsampling by an integer factor is generally described as a two-step process, where first a sequence is expanded by introduction of empty values between existing readings and interpolating the signal [12]. The first step is already complete, since it is the actual problem that needs solving. There are many methods for achieving the second step. In the domain of image upscaling the most basic method is nearest neighbour, essentially KNN with $K = 1$ [13]. More complex method is bilinear interpolation and bicubic interpolation, which is linear interpolation and cubic interpolation in two dimensions respectively [14]. Local regression techniques can also be utilized, such as Locally Weighted Scatterplot Smoothing (LOWESS) [15], although this method is more used for smoothing out data, and not upscaling or data imputation.

At this stage it is important to mention the problem of information leakage [16]. It is especially important in ML tasks to avoid *receiving information from the future*, especially when working with data series and creating artificial training

and testing datasets. All imputation processes should be performed on the portion of the data that is considered available at a given time.

In the following section, and consequently in the related publication [AT1], only the most basic imputation algorithms are taken into consideration - forward filling (last valid value carried forward), and linear interpolation. This selection was done due to lack of systematic comparison of such basic methods, as well as a working assumption that some signals will be better infilled using one or the other method. To cite a recent review of ROP prediction publications [17] on the problem of data gaps: "*In general, this problem was omitted*".

Methods for small gaps

When dealing with data series one of the most common filling methods is forward filling, otherwise known as *last observation carried forward* [1]. In this basic method the series are processed from the beginning, and when value at position n is missing, then the value from previous position, $n-1$, is used in its place. This method allows to populate all the empty cells, except for the situation when the first value is missing; in such edge case operation of analogous backward filling may be performed after forward filling.

Another basic method of handling missing values that is applicable for data series is linear interpolation, which is a curve fitting method. When utilized for imputation it is a robust method that for a gap spanning from position n to m, henceforth designated y_n and y_m respectively, requires to know only the values y_{n-1}, y_{m+1}, and the index value for these datapoints such as measured depth, here designated: x_{n-1}, and x_{m+1}. This allows for infilling of any gaps, except of the edge cases when first and/or the last value in the dataset is missing. This dissertation takes into account only those basic infilling algorithms for two reasons: they work on vast majority of the gap sizes, and the focus of presented research was to evaluate the principle of varied and semi-automatic selection of gap filling algorithm, and not on achieving lowest possible error.

General equation for linear interpolation is given as:

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0} \tag{2.2}$$

Solved for y it becomes:

$$y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0} = \frac{y_0(x_1 - x) + y_1(x - x_0)}{x_1 - x_0} \tag{2.3}$$

Error of the linear interpolation can be estimated using Rolles theorem, which was used extensively in the past when sparse tabulated data was more common [18]. It utilizes the maximum value of a second derivative of the function at hand, assuming the second derivative is continuous.

$$|R_T| \leq \frac{(x_1 - x_0)^2}{8} \max_{x_0 \leq x \leq x_1} |f''(x)| \tag{2.4}$$

Where $|R_T|$ is the maximum error calculated per Rolles theorem and f is the evaluated function. While for tabulated data it will necessarily be only an approximation, it is a useful tool when quantifying the introduced error.

Choosing between forward filling and linear interpolation automatically

Recent review of or ROP prediction publications identified that data gaps are a problem generally omitted [17]. While forward filling and linear interpolation are common algorithms, I have not identified a publication that would discuss the differences between them in the context of machine learning applied to drilling.

An algorithm was proposed and published [AT1] to select whether a given attribute should be infilled with forward filling or linear interpolation. A dataset with an attribute X containing $k + 1$ data points defined as

$$X = \{x_0, x_1, \ldots, x_k\}. \tag{2.5}$$

A new dataset can be calculated using simple partial derivative, using dot notation:

$$\Delta X = \{\dot{x}_0, \dot{x}_1, \ldots, \dot{x}_{k-1}\} \tag{2.6}$$

and n_o is the quantity of zero-valued of data points in ΔX. The proposed solution is to apply algorithm:

- Forward filling if $\frac{n_o}{k} \geq \theta_{th}\%$
- Linear interpolation if $\frac{n_o}{k} < \theta_{th}\%$

A threshold $\theta_{th}\%$ is a hyperparameter that can be tweaked to achieve the best results in the ML tuning process.

Two case studies were run based on ML model proposed in published work [AT1], described in section 3.5.1. Utilizing F9a, depth base dataset, prediction of inclination and prediction of ROP was performed in two independent studies. The accuracy of the prediction was calculated using MAE and training-while-drilling [AT4], a continuous learning approach [19], described in detail in this dissertation in section 3.4.

Two case studies are presented in publication [AT1] and there is a statistically significant difference in MAE depending on the threshold parameter. While the difference is not very big, it is approximately 6.5% between best and worst case in the inclination case study, as well as more modest 1.6% for the ROP case study. This is most likely due to fact that the inclination case study depends to a much higher degree on the temporal effects then the ROP one. Note that the gap filling algorithms are compared via a case study, since no ground truth for the data within the gaps exist that would allow for direct comparison. Furthermore, comparison of error in a case study provides results in terms of practical prediction error reduction.

More work is needed to identify the exact reason in performance difference between the infilling using forward filling and linear interpolation. One of the

main difficulties is that the ML model itself at this level of complexity is, for all intents and purposes, a black-box device where it is not clear why it behaves the way it does. Nevertheless, there is a difference in performance depending on which algorithm is used to what extent and it is a worthwhile hyperparameter to tune. It is also worth noting that it is likely that specific attributes work best with a specific algorithm (f.ex. ROP always performs best with linear interpolation), and further research is required to identify and confirm such settings.

2.2.2 Data resampling methods

As discussed in earlier sections, real-time drilling logs are in practice a combination of various logging systems that work at different frequencies, and provide data at different times. This necessarily creates a log where, after merging all the data sources, the sampling rate is not even. There is also an additional reason unique to drilling logs for the sampling rate to vary even more. While logging is essentially always done in the time domain, drilling logs are often converted to be indexed along the measured depth, making so called depth-based data. There are a number of reasons for that, such as:

1. Drilling operation consists of multiple technical elements that are irrelevant for many purposes, like adding pipe or tripping out to relocate heavy weight drill pipe, where the drill bit is not on the bottom of the well.
2. Equipment failure or unfavourable weather may stop drilling for an extended period of time
3. It is much easier to analyze one seemingly uninterrupted drilling process
4. It often makes more sense to perform predictive analysis in terms of depth, not time.

When converting time based logs to a depth base, the sampling rate for sections with high ROP will be lower, while sampling rate for areas with low ROP will

be higher. At the extreme, areas where no progress is being made will create multiple data points at the same depth.

Research presented in this dissertation was based on depth-based data.

Why is resampling necessary?

Resampling, or more precisely making the data points equidistant, is a requirement for many ML models that contain temporal, or dynamic information. Consider Figure 2.5, where a sine wave is evaluated. Dashed *black dashed line* is the ground truth. *Black dots*, are data points that were selected from a random, normal distribution for $x \in (0, 10)$. If this data is however plotted (*red diamonds*) without the index information at equal distances the sine wave becomes significantly deformed. This makes it much more difficult for a ML model to learn the pattern existing in data. To solve this problem data can be resampled. Here, using a K-Nearest-Neighbours algorithm [20, 21], new dataset was created and was plotted equidistant along x axis as *blue X's*.

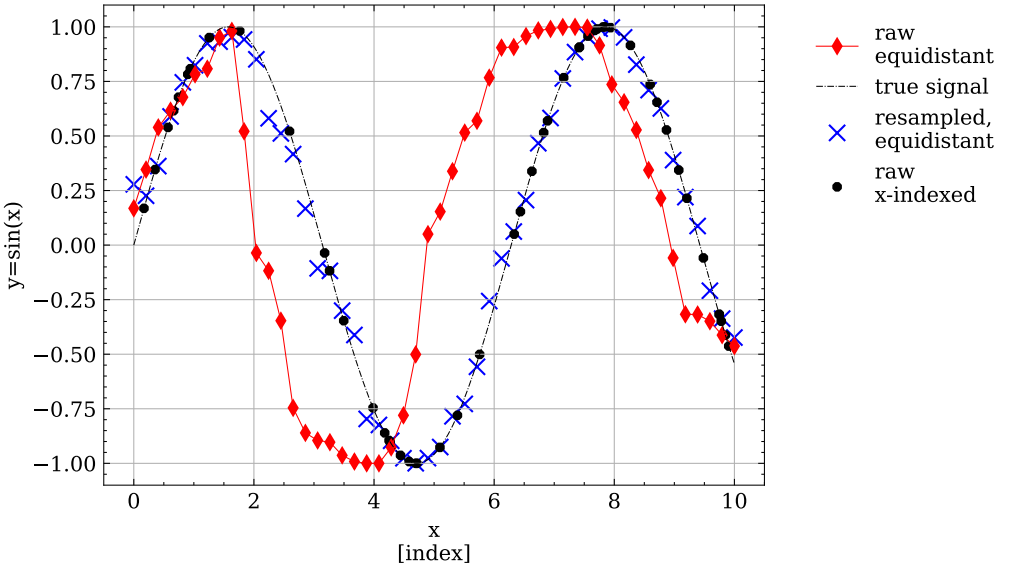


Figure 2.5: Importance of resampling. Effects of unevenly sampled signal.

In general, established RNNs such as LSTM and GRU do not take into account index. There is recent research into feeding irregularly sampled data directly into ML models [22, 23, 24], however solving the problem by modifying the data allows for use of well established techniques and algorithms.

Literature review

Resampling, in case of non-uniformly sampled signals in practice requires creation a regression function that can provide a predicted value of the signal at hand on the arbitrary index position. It is worth noting that in machine learning *resampling* often refers to correcting imbalanced datasets; this is often done in classification problems to make various classes are equally represented [25].

A thorough, albeit dated overview of the problem of sampling at uneven rates [26]; it describes various methods of dealing with the issue, notably linear interpolation, cubic interpolation, boxcar averaging, equivalent to RNR algorithm uniformly weighted, and weighted average, equivalent to RNR with distance weights. In drilling domain only one earlier publication was identified that implemented a form of resampling, here cubic spline interpolation [27].

As the topic of resampling is fairly unexplored, for a thorough investigation two algorithms were selected, K-Nearest Neighbours [13] as well as Radius Neighbour Regressor [28, 29]. These two algorithms have a limited number of parameters to set, K-neighbour value and radius respectively, as well as allow for alternative weighting of datapoints. They are quick, easy to understand, and do not generate artifacts such as Runge's phenomenon in polynomial regression [30] that would require additional verification of results.

The algorithms

Two algorithms useful for resampling were evaluated in detail: K-Nearest Neighbors (KNN) [31], and Radius Neighbor Regressor (RNR) [28].

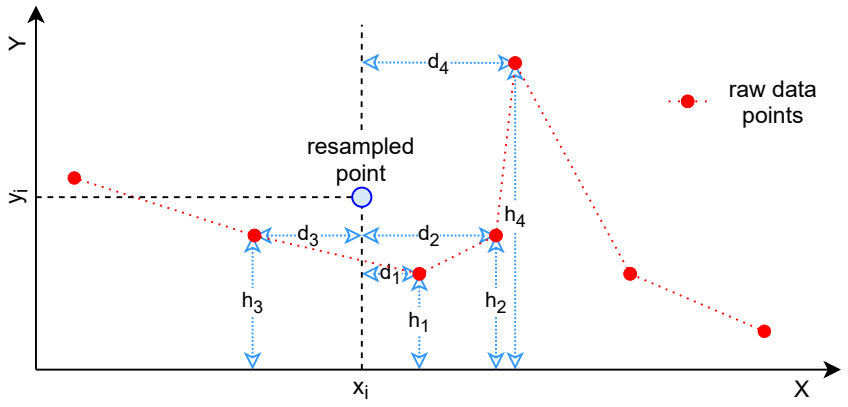


Figure 2.6: KNN algorithm

KNN algorithm is conceptually presented in Figure 2.6. Parameter K is set to 4, hence 4 data points nearest to the desired location x_i are found. Following calculation is done to establish point y_i:

$$y_i = \frac{1}{K} \sum_{i=1}^{K} h_i \tag{2.7}$$

Where h_i is the datapoint value as sorted from closest to the desired location x_i.

Alternatively it is possible to add a distance weight elements to make the points closest to the resampled point more important than the further ones as:

$$w_i = \frac{1}{d_i} \tag{2.8}$$

where d_i is the distance of the i-th point to the desired location x_i.

Alternative, albeit similar algorithm, is RNR is shown in Figure 2.7. The difference compared to KNN is that the inclusion of the points for resampling is based on the radius r and the distance from the new data point x_i, hence the quantity of them will vary throughout the dataset. Value y_i is calculated the same way as in equation (2.7). The same as with KNN distance based weights can be

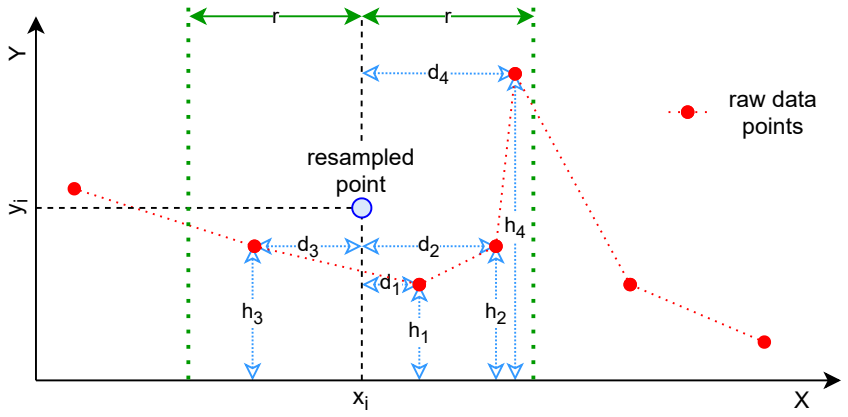


Figure 2.7: RNR algorithm

utilized. Using RNR there is a risk that no point from the original raw dataset is within the specified radius. That is why throughout the research presented in this dissertation radius is defined using the multiples of maximum distance between the existing datapoints.

Side effects of resampling

Like with all data processing techniques there is a risk of side effects. Since resampling is in many aspects similar to a centered moving average the same problems of smoothing out the signal may surface. Exploration of various resampling algorithms with different settings is performed in one of the publications attached to this dissertation [AT1].

If the deformed signal is left unnoticed, it may cause an increase in the model performance that would not be representative of the real-life predictive quality. At the extreme, when using KNN the K parameter is set to be equal to sample count in a dataset, the resampled dataset will contain of attributes of constant value and undoubtedly perfect apparent performance. Furthermore, when resampling with a high radius (or equivalent K value) may be detrimental to real-time applications. If a model is trained on data resampled with $r = 5m$ (ref. Figure 2.7), then in real-time operation only data $5m$ away from the bit will be reliable, since in

resampling radius is in both forwards and backwards directions in relation to the index.

Root Mean Riemann Squared

The base for the development of new resampling quality is the previously mentioned work using sum of absolute differences [32]; following changes to the method were introduced in [AT1]:

- difference would be squared to penalize high deviations
- average value instead of a sum would be used to make the metric independent of the data length.
- root of the sum will be utilized, bringing the method in line with RMS calculations

There is additional practical problem to be solved however: original and resampled datasets to not share the same indexes and therefore there is no straightforward difference to be calculated between two points. In practice, what needs to be calculated is the average squared distance between two piecewise functions, defined as straight lines between the consecutive datapoints. This piecewise function is here designated as $g(x)$. This can be calculated using a Riemann sum [33]. This results in an equation that is called Root Mean Riemann Squared (RMRS) in the publication [AT1]:

$$RMRS_g = \sqrt{\frac{1}{u}\sum_{i=1}^{u}\left(g(x_i)\right)^2}. \tag{2.9}$$

where u is the quantity of divisions as understood in a classical Riemann Sum.

Practical application of RMRS to optimize resampling.

RMRS was applied to select the best resampling algorithm on a case study of a Volve dataset well F9a, depth based dataset. Sampling rate was selected at

0.25m, in line with the research done on that well for the inclination prediction [AT3, AT4]. Two algorithms were tested, KNN and RNR, both with uniform weighted samples and distance weighted samples. Parameters varied were the K, the neighbor count for KNN and radius multiplier for the RNR, where maximum distance between datapoints was checked in the dataset and that value was multiplied. This is explored in detail in [AT1].

2.3 Benchmarking datasets

Recent survey revealed that 70% of scientists attempted and failed to reproduce results from a publication [34] touting reproducibility crisis in science. As a part of publication [AT2] there was a review performed of publications that focused on ROP prediction [35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]. Vast majority of them did not share data at all. Two publications [48, 49] used ancient by modern standards data, which is 30 samples from 1974, with data listed verbatim in references [50]. There was also a publication listing all 25 samples used [51] which were later re-used [52]. This is a very poor data landscape to benchmark novel methods.

To tackle this problem a reference dataset was created [AT2] for the purposes of benchmarking ROP prediction methods. It is based on the Volve dataset [6] and retains the original open license applicable at the time: CC BY-SA-NC 4.0[1]. It consists of 7 wells from the original dataset and are depth-based.

Additional problem in ML model benchmarking is that different researchers apply their models differently, not always in a correct manner. Three scenarios were defined in [AT2]: Continuous Learning [19], All for One, and One for All, that define how one is to divide the data into available and for testing. Available data can be used for training and validation purposes. The scenarios are visualized in Figure 2.8. Those scenarios were designed to represent real-life application

[1]In September 2020 Equinor changed the license to their own, equally open Equinor Open Data Licence

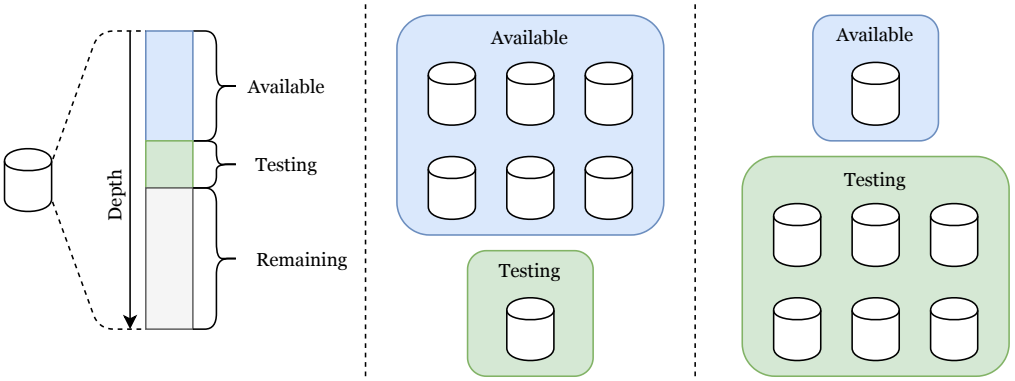


Figure 2.8: USROP recommended scenarios, from [AT2]

possibilities, that is one reference well used for training the model that is deployed to multiple wells (One for All); multiple wells can be available for making the model while one is left for testing, evaluating the impact of additional data available (All for One); lastly the continuous learning approach, where no pre-trained model is available as it is being trained while drilling. Technical details on how perform this study is included in the related publication [AT2].

The publication also suggests the error metric, as there are many to choose from. Weighted Mean Absolute Percentage Error (WMAPE) is recommended as more intuitively understandable by petroleum experts than R^2 more commonly seen in scientific publications:

$$WMAPE = \frac{\sum_{t=1}^{n} |y_t - \hat{y}_t|}{\sum_{t=1}^{n} |y_t|} \tag{2.10}$$

where y_t is the true value and $\hat{y}_t$ is predicted value, with n being the dataset length.

Having all these boundary conditions established the publication shows results for ROP prediction using a number of off-the-shelf algorithms, as well as physics based Bingham model [53], to establish base performance. Publication [AT2]

presents reference results for the all the described scenarios using various metric methods. The publication also shares the code used to build the dataset from the Volve data.

Chapter 3

Machine learning modelling procedure

3.1 Introduction to Machine Learning

Machine Learning is a method where algorithms develop relationships between provided input and output data, called training, so that afterwards the output can be predicted for an arbitrary input. This is best understood on the most basic example. When data is plotted in a typical spreadsheet program such as Excel, there is an option to add a trendline. This is one of the most basic examples of machine learning. The equation provided by the software allows to estimate the total payment per arbitrary number of claims.

While there are multiple ML algorithms the one that is the basis of the work contained in this dissertation is an artificial neuron. It is based on a biological neuron and was developed in 1943. Currently employed basic artificial neuron structure is shown in Figure 3.1.

The input vector containing values from x_1 to x_n is multiplied by weights from w_1 to w_n. The result is then summed (Σ) with bias value. This sum is then passed through an activation function, here designated as φ. There are multiple

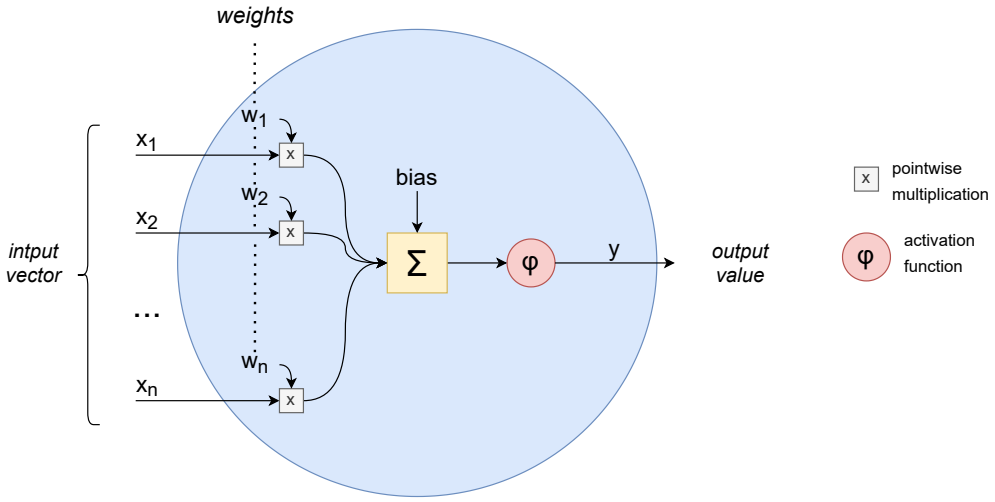


Figure 3.1: Structure of the artificial neuron.

activation functions in use: linear, rectified linear (ReLU), sigmoid, hyperbolic tangent (tanh), and many more. The output of the activation function is the output of the neuron, designated here as y.

3.2 Data split in data series

In ML the dataset at hand is in principle split into three separate portions, training, validation and testing [54]. The *training* dataset is used to establish internal coefficients and structure of the ML algorithm. It provides the algorithm both the *input* and the desired *output* in a goal of establishing a correlation between the two. *Validation* dataset is a portion of data that is used for hyperparameter tuning, or more generally, selecting the best algorithm to use. During development of the ML model, the algorithm is presented with the input from the validation dataset, and based on the "learning" of the data gained from the training dataset it provides the estimate of the output. The difference between the true output of the validation data and the value predicted by the ML algorithm is used to adjust, manually or automatically, the way the model is trained. Lastly, the testing dataset in principle should be evaluated only once at the end of the ML

model development process, to provide the final score by calculating the difference between the true output of the test dataset and the ML predicted output. No further modifications to the ML model are allowed at this point, since otherwise the test dataset becomes just another validation dataset. In principle it is not necessary to use a strict validation dataset; the main split of the data is into training and testing, with the validation data being taken from training as needed.

While there are discussions on the methodology itself [55, 56] there are two main methods of splitting data at hand into training and testing. One is a random data split, where samples are allocated to one or the other dataset at random at predefined quantities, and sequential split, applicable to data series, where at one point a cutoff exists, splitting the data into training and testing at a given index value, be it time, depth, or other.

An experiment was performed where two random walks are generated and a machine learning algorithm is employed to predict one based on the other [AT2]. In this clearly nonsensical dataset, with two random series, when train/test split is performed randomly a basic off-the-shelf ML algorithm, Gradient Boosting Regressor of scikit-learn library [21] achieves impressive result of $R^2 = 0.946$. By tweaking parameters such as the split ratio and random seed it was possible to predict one random walk based on other random walk with $R^2 = 1.00$, when rounding to two decimals. This is clearly an absurd result.

Proper way of splitting data in series is to do so in continuous sections, where data up to a given point is considered training, and data beyond that point is used for testing. This way of splitting a dataset is at the core of the *training-while-drilling* approach presented in the next section.

Model validation

Part of the available data, the training dataset, can be set aside as validation data, somewhat reducing the size of the training set. Work presented in this dissertation [AT4, AT3] reduces that set typically by the last 20% to evaluate

the model while training; this validation set is the newest data which showed the best results later compared to picking early, middle, or random samples.

Having validation dataset it is possible to implement *early stopping* [57], a technique where validation error is constantly monitored. Model with a lowest validation error is retained, and as long as consecutive epochs yield reduced validation error, this model gets updated. At one point model will begin to overfit; this is a moment where while training error will continue to drop, the validation error will begin to increase. Training will continue for a number of epochs defined as *patience*, in case of model shown in this dissertation it is set to 25. In that time either the validation error will reach a new minimum, in which case the patience counter will be reset to original value, or the training will stop. This is shown schematically in Figure 3.2.

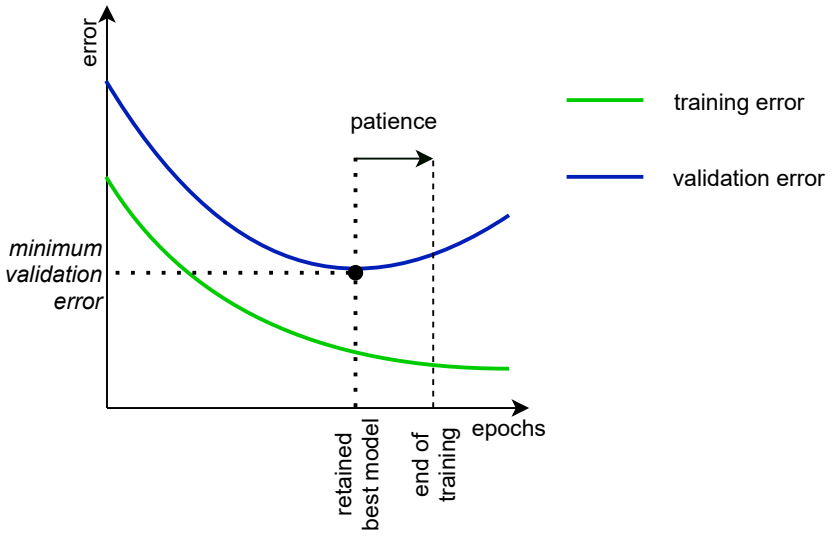


Figure 3.2: Concept of early stopping in machine learning training

3.2.1 Feature engineering

To make the job of the ML algorithm easier it is often beneficial to perform *feature engineering*. This is a process in which expert domain knowledge is used to apply mathematical transformations to existing inputs so that additional input

is created that has a more straight-forward correlation with the output, making the prediction easier, and therefore, error lower.

The term feature engineering was coined approximately 20 years ago [58] to establish it as a part of a software lifecycle. A basic example of feature engineering would be deconstruction of a date. While ML algorithms can work with formats such as Unix time, there is a lot of potentially useful information that can be mode more easily accessible. Hour of the day, day of the week, a binary attribute for "is it weekend?" can make ML problems such as road traffic prediction much more successful.

Feature engineering is widely adopted in drilling related problems [36, 59, 38]; a recent publication explores feature engineering as a key element in predicting ROP, where DWOB values are corrected through physics based models achieving significant performance increase [AT8].

In the case study of inclination prediction feature engineering was applied to the target signal, the inclination, explained in detail in related publication [AT3]. The rationale is that directional drilling process is to a high degree independent in its behaviour of the absolute inclination; therefore the output was moved to a local coordinate system. For each input sample inclination starts at x_{i-n} and ends at x_{i-1}, and then continues as output at x_i and ends at x_{i+m}. Scaling was applied, where inclination at x_{i-n} is equal to 0, and at x_{i+m} it is equal to approximately 1. It is not guarantied to end at one, because within a sample the inclination can rise a lot, or not at all, and scaling uniform throughout all samples is necessary. The process is visualized in Figure 3.3

Such transformation allows the network to generalize better and be trained on a single well. It abstracts the inclination prediction process so that the absolute inclination is irrelevant. Scaling in range $(0, 1)$ was selected as it is standard procedure in ML. There are studies comparing different scaling methods, standardization, normalization and other algorithms, with to single algorithm being clearly better than the other [60].

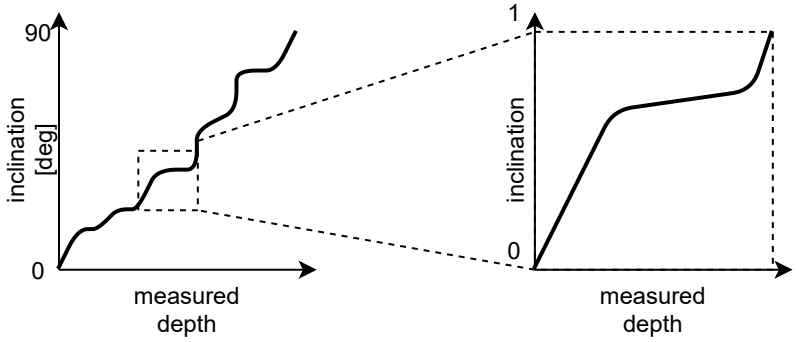


Figure 3.3: Local coordinate system transformation of inclination

3.2.2 Attribute selection

While feature engineering creates new attributes based on the existing ones, the attribute selection focuses on picking the most useful attributes from the ones that are available. Recent publication performed systematic analysis of various attribute selection techniques [61]. Three basic categories of feature selection methods were identified:

- Intrinsic
- Wrapper
- Filters

Intrinsic methods are a part of the ML process itself; for example in Decision Tree algorithm the less important features will simply not be a part of the resultant tree, effectively omitting some of them. *Wrapper* methods perform a search for a subset of features that will perform best. An example of such algorithm is Recursive Feature Elimination (RFE) [62], where an ML algorithm is trained on the initial set of features, and the least important feature is removed. This process is repeated until a desired quantity of features is remaining.

Filters category is very broad, and focuses on mathematical evaluation of the relationship between the features and the target. The aforementioned publication

[61] categorizes various algorithms based on their applicability using the input-output data type matrix. It is necessary to identify if the input is numerical or categorical, and whether the output is numerical or categorical. In relation to drilling related ML problems explored in this dissertation both inputs and outputs are numerical. For such combination Pearson correlation coefficient [63] and Spearman's rank correlation [64] are listed as the main algorithms of interest, which are often compared in literature [64] as they are closely related. Pearson's method is based on evaluating the linear correlation between the attributes, being equal to the ratio of covariance of the attributes to the product of standard deviations of the attributes:

$$\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{3.1}$$

where $\rho_{X,Y}$ designates the Pearson's coefficient between variables X and Y, *cov* is the covariance value of two attributes, and Σ stands for standard deviation.

Spearman's rank correlation is defined similarly, where actual values are replaced by their ranks (position) in the dataset:

$$r_s = \frac{\mathrm{cov}(\mathrm{R}(X),\mathrm{R}(Y))}{\sigma_{\mathrm{R}(X)}\sigma_{\mathrm{R}(Y)}}, \tag{3.2}$$

where r_S is the Spearman's rank, and $R(\cdot)$ designates the rank (position) in the dataset.

There are newer methods for numeric input to numeric output correlations. One of such algorithms is called Predictive Power Score [65]. It utilizes Decision Trees to evaluate if given attributes A can predict attribute B. One of the major touted benefits of this approach is that it makes the correlation matrix asymmetrical, as many correlations are in reality. An example of asymmetric correlation is post codes (zip codes) and cities. Any given numeric post code will be directly correlated with a city, for example 75005 can be reliably decoded to Paris. However this correlation does not hold true in the other direction, as

city of Paris has multiple post codes and pinpointing the correct one for a given sample requires more data.

An alternative to feature selection is *feature projection*, where original inputs are transformed into a different set of inputs. One of common feature projection methods in use is Priciple Component Analysis (PCA) [66]. This method decomposes multi-dimensional dataset into principal components; these are eigenvectors of the multi-dimensional data's covariance matrix. The components of the PCA transformation are sorted based on the quantity of variance explained, with all components explaining 100% of the original variance. It is the components that carry the most variance information that are desirable in retaining as inputs to ML algorithms, while those explaining a small range of variance can be omitted.

It is important to highlight that while feature selection algorithms simply select attributes that may be the most valuable for the ML algorithm, the PCA transformation completely changes the dataset making it unrecognizable to the naked eye. Another key consideration is that the PCA transformation, similarly to ML algorithm is *trained*, or here *calculated*, on a dataset, based on which eigenvectors and eigenvalues are established. These create a core of the PCA transformation allowing translation between the raw values to PCA components and back, albeit with data loss unless all PCA components are used. This allows for calculating PCA transformation of previously unseen data.

Alternative feature projection methods such as Non-negative matrix factorization (NMF), Linear discriminant analysis (LDA), Generalized discriminant analysis (GDA), Autoencoders, and T-distributed Stochastic Neighbor Embedding (t-SNE) are out of scope of this dissertation.

In drilling

Many drilling related publications utilize Pearson correlation coefficient [63] to identify attributes correlated with the target output. This strategy works best for simpler problems and simpler models, as it basically uncovers correlation close to

linear, and therefore it is not useful for cyclic signals. Furthermore, more complex ML models are in principle deployed where relationships far beyond linear exist.

Another approach for selecting inputs for the ML model is expert knowledge. It may be the case that highly correlated attributes contain certain information that is simply duplicated, while less correlated signals carry more nuanced data. It is inadvisable to utilize all available attributes as inputs, since there may be literally hundreds of attributes recorded in drilling logs [AT6] which will likely yield good results in training, but high error in testing, as spurious correlations will be used by the ML model. This is particularly visible when limited data is available, as in proposed training-while-drilling approach in further chapters and the relevant publication [AT4].

Alternatively, PCA can be utilized as dimensionality reduction technique [1]. Eigenvectors are calculated based on training data only, while transformation is applied to both training and testing datasets. This technique showed to perform very well in the presented case study [AT1], where 5 components showed lowest error either significantly outperforming selection based on Pearson coefficient and ppscore [65], or performing approximately on-par, depending on the exact case study setup.

3.3 Recurrent Neural Networks

There is a vast number of algorithms that fall under machine learning; research presented in this dissertation utilizes Recurrent Neural Networks (RNN) [3] that are able to capture the dynamic behaviour of the system by using past values as inputs to predict the future values, outputs.

RNNs, or rather their modern implementation using LSTM and GRU, play a significant role in many ML applications. While RNN is theoretically Turing Complete [67], meaning it can be applied to any computational problem, they excel in a specific sets of tasks. They are a common choice for language translation, or sequence to sequence learning [68]. LSTM is also widely used in both

handwriting recognition as well as speech recognition [69]. There are also more exotic applications of LSTM; in 2016 Google's OpenAI created a deep neural network to play a popular computer game Dota 2.

What LSTM is mostly known for and widely applied in is time series forecasting, which is also the area that is most relevant to real-time drilling forecasting. LSTM in such problems hugely outperforms non-ML approaches such as ARMIA [70].

In 2014 a simplification of LSTM was introduced, a Gated Recurrent Unit (GRU) [71]. The key practical difference betwen GRU and LSTM is that the former tends to perform better in certain small datasets [72, 73]. This becomes particularly relevant in proposed mode of model deployment presented in this dissertation called training-while-drilling [AT4], where drilling operation begins with blank, untrained model, and as the data is being collected, the model begins to perform better and better. Additional benefit of GRU is that it is similar to LSTM, where the models can be switched back and forth during development phase without significant changes to the software.

Variants of RNN are used in petroleum based problems, but likely due to their more advanced nature are less common than basic ML algorithms. There is research into ROP prediction [42] as well as gas kick prediction [74] using LSTM. GRUs are also used, for example in tool condition monitoring [75]. Both architectures were employed for gamma log prediction [76] with mixed results.

The main benefit of RNN-like architecture is that the sequence of the input has meaning. If n time-steps are used as input to an ML algorithm, the information about sequence of the input is lost if MLP, Decision Tree, or SVM is used. While through the training process this information can be learned, matching the sequential nature of the data to the architecture of the algorithm allows for much better performance, similarly to how feature engineering boosts the ML scores even though application of basic math. On the other hand, RNN architectures will struggle with non-sequential data, as they would have to effectively learn *against their own nature.*

3.3.1 RNN cell structure

As visualized in Figure 3.4 the cell A takes not only input x_t but also a value v_{t-1}, the recurrent connection generated in the previous step. That connection may contain the generated output from the previous step, h_{t-1}, but depending on the exact cell architecture it can contain additional data. This process generates output h_t. For practical purposes this process is typically considered in the unfolded state, as shown on the right hand side of the Figure 3.4.

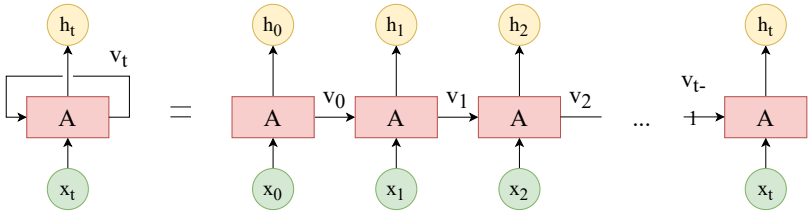


Figure 3.4: Recurrent Neural Network [3]

Internal structure of the basic RNN cell is very simple. Current input vector and previous output vector are concatenated and processed through a tanh layer. Figure 3.5 provides visualization of the RNN cell structure; note that the structure is somewhat simplified and omits basic elements of an artificial neuron.

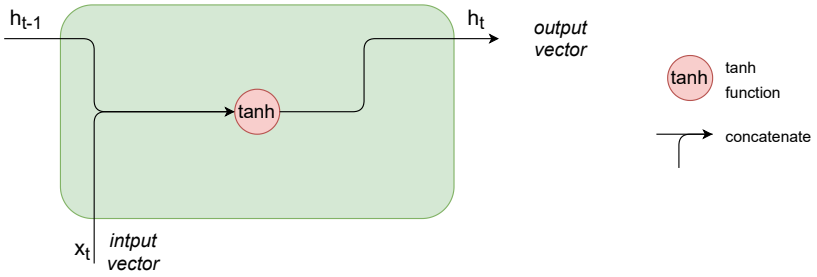


Figure 3.5: RNN cell structure [3]

Mathematically it can be described as:

$$h_t = \sigma(W \cdot x_t + U \cdot h_{t-1} + b) \tag{3.3}$$

where h_t is the output vector, σ is the activation function, in Figure 3.5 hyperbolic tangent function is listed. W and U are weights for the input and output vectors respectively, and b is bias.

3.3.2 Long Short-Term Memory

To solve a number of practical problems in training RNNs a new cell structure was developed called Long Short-Term Memory (LSTM) [77]. It has a significantly more complex structure than a classical RNN cell, as seen in Figure 3.6. The architecture of the LSTM cell is made such that it facilitates remembering information for longer periods of time.

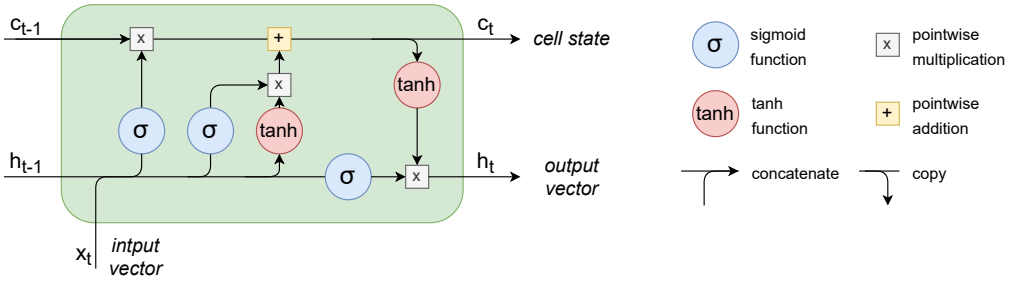


Figure 3.6: LSTM cell structure [77]

The cell structure can be expressed mathematically as shown below:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{3.4}$$

The f_t is the activation vector of the forget gate. W and U designate matrices that are the weights of the input and recurrent connections with subscripts identifying the specific vector. σ is the activation function with subscript identifying the relevant vector. For LSTM σ_g is sigmoid function, σ_c is hyperbolic tangent function, and σ_h is also hyperbolic tangent function. t subscript identifies the current time-step, and analogously $t-1$ the previous time-step. x_t is the input vector and b is bias, separate for each relevant vector.

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{3.5}$$

The i_t is the activation vector of the input gate

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{3.6}$$

The o_t is the activation vector of the output gate

$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{3.7}$$

The $\tilde{c}_t$ is the activation vector of cell input

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \tag{3.8}$$

The c_t is cell state vector

$$h_t = o_t \circ \sigma_h(c_t) \tag{3.9}$$

The h_t is the output vector of the LSTM unit. The $\circ$ symbol designates an pointwise multiplication, element-wise product, or Hadamard product, where two matrices A and B of same shape are the input and the output is a matrix of the same shape C, where elements are calculated as $c_{ij} = a_{ij} b_{ij}$.

3.3.3 Gated Recurrent Unit

Active research into RNNs yielded a structure called Gated Recurrent Unit (GRU) [71] which is based on the LSTM, but has fewer elements, as seen in Figure 3.7. This structure was of interesting for the presented research as it may perform better than LSTM when applied to smaller datasets [72, 73], which was the case

for training-while-drilling approach [AT4], where an ML algorithm is trained on data from the well where it is applied at the same time.

Expressing the GRU structure mathematically:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{3.10}$$

where z_t is update gate vector. W and U designate matrices that are the weights of the input and recurrent connections with subscripts identifying the specific vector. h_{t-1} is the output vector of the previous state, b designates bias with subscript designating the relevant vector. x_t is the input vector. σ_g is the activation function, here sigmoid function.

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{3.11}$$

r_t is the reset gate vector

$$\hat{h}_t = \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \tag{3.12}$$

$\hat{h}_t$ is the candidate activation vector, σ_h is the activation function, here hyperbolic tangent function. r_t is the reset gate vector.

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \hat{h}_t \tag{3.13}$$

h_t is the output vector and z_t is the update gate vector.

3.4 Training-while-drilling approach

One of the challenges in machine learning adoption in the petroleum industry is the fact that very often wells will differ in lithology and equipment used, causing

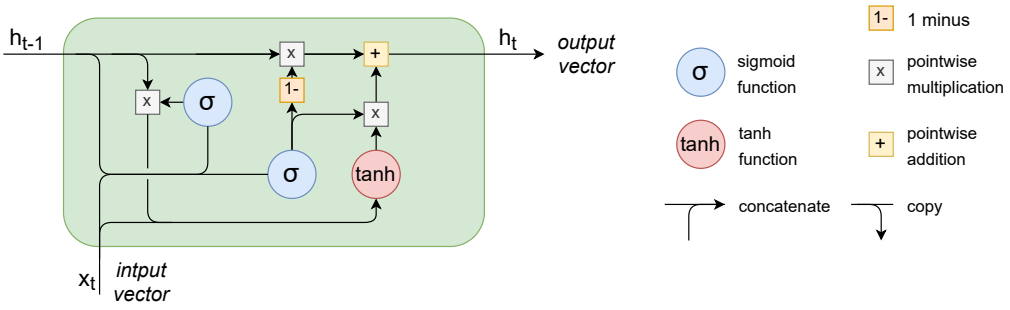


Figure 3.7: GRU cell structure [71]

big differences in system response. This can cause machine learning models trained on one well perform poorly on other wells. Furthermore, if a piece of equipment fails it may be decided to finish drilling using a different technology, complicating matters even further.

It is however not necessary to train machine learning model on a different well than it is being applied to. Continuous learning paradigm [19] discusses ML models that are continuously updated, so that new *experiences* can influence predictions made in immediate future. This method, applied to drilling was called training-while-drilling [AT4], where drilling starts with an untrained model, and as the data is collected it is being used for training. While reusing data from the same well to perform predictions has been done recently [36, 39], research performed for this dissertation was the first application of truly continuous learning at high resolution [AT4].

Training dataset in training-while-drilling approach is considered to be in $[0, x\%]$ of the available data, and the testing dataset is in $[x\%, (x + F)\%]$. The value x is varied from as close to zero as possible, where there is sufficient data to train a model, to value near 100%. This describes the increasing measured depth while drilling. The F value drives the size of the testing dataset, which can be in practice equivalent to the length of MD drilled while the new model is being trained. The complete process is described in detail in publication [AT4].

It is important to compare new methods to the previously used ones. In the publication [AT4] total four approaches were evaluated:

1. local coordinate system method, where inclination is moved to the point (0,0) for the first point in the sample. This is a feature engineering approach, ref. Section 3.2.1. A branched MLP+RNN neural network design was used, see Section 3.5.1.

2. incremental method - inclination value was converted to inclination change, again a feature engineering approach, and later recovered through cumulative sum. The same branched neural network was used as before.

3. MLP method - off the shelf Multi-Layer Perceptron based off an artificial neuron (ref. Section 3.1) was used to predict inclination change. This is a purely row-wise application and no temporal information is included in the model

4. XGB method - an off the shelf algorithm utilizing extreme gradient boosting [78] was applied row-wide, without temporal information. This is a very popular multi-purpose algorithm and therefore was chosen for this benchmarking task.

Results are presented in the publication [AT4], where RNN based methods significantly outperform MLP and XGB. One of the key findings of that publication is that to achieve best performance two different approaches have to be used, since using inclination change performs better for the shorter horizon prediction, while the local coordinate system approach performs better for further predictions, as it does not accumulate error - the incremental method contains a cumulative sum to restore actual inclination, hence sums the error from all earlier predictions.

3.5 Case study of directional drilling

Main case study used throughout the research presented in this dissertation is related to directional drilling, explained shortly in Chapter 2. One of the practical problems in directional drilling is the distance between the DNI sensor and the drill bit. Especially in the case of the old and reliable bent sub drilling method

this distance can be very significant; schematics included in the Volve dataset show it being 23 meters behind the bit. This results in a problem here called *sensor lag*, where a portion of a well is drilled, but it is yet to be measured by the sensor. The targets that directional drillers have to hit are often very narrow, for example in the Bakken field pay zones are just between 5m and 15m thick [79]. This makes providers of directional drilling equipment compete in terms of putting the sensors as close to the bit as possible; this is a big technical challenge, since front of the drilling tool is occupied by complex mechanisms that allow the BHA to turn. Due to high pressure, temperature and vibration levels the sensors suitable for downhole use are much bigger in size than the ones found in modern consumer electronics. This problem was also presented for the Forsker Grand Prix 2020 [AT9].

The resulting sensor lag makes directional data lag behind the real-time attributes. This is shown in Figure 3.8; to the right from the sensor exists a portion of the well where attributes such as WOB, RPM, ROP, SPP, etc. were collected while drilling and were available for processing and analysis, for all intents and purposes, instantly. Directional information such as inclination and azimuth however, are available only from the portion of the well directly behind the sensor, resulting in delayed reaction to the actual direction of the well drilled. The goal of the case study is to predict the directional information in the section where it is missing.

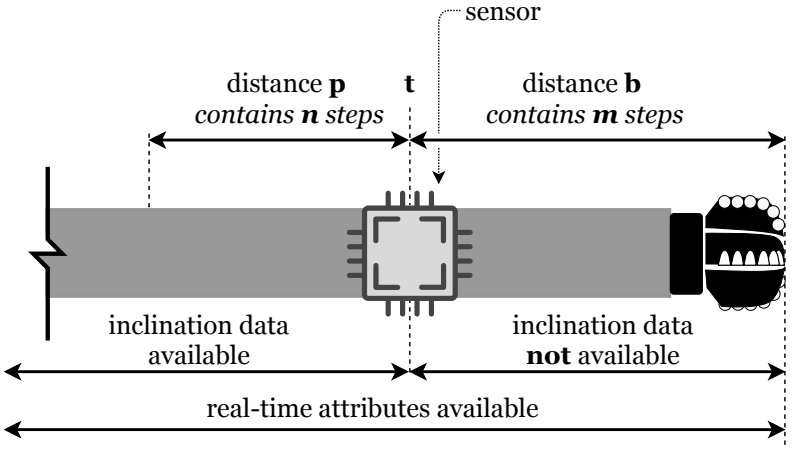


Figure 3.8: Sensor lag, originally in [AT4]

3.5.1 Branched model architecture

To achieve high quality prediction when using ML techniques correct inputs and correct algorithm have to be selected. In principle, the more directly relevant information is provided to a ML model, the better the result quality can be expected afterwards. Just like with human brain, where different areas of the brain are wired differently for different tasks, such as occipital lobe handling most of the visual stimuli [80], it is important to pick the ML algorithm best suited for the task.

To predict the lagging inclination data, as described in the described case study before, there are two distinct methods of prediction that can be used:

1. Predicting when *sliding* and *rotating* section of the well are based on ROP, WOB, Torque, etc.
2. Predicting the characteristic rising and holding pattern of the inclination

While these two elements sound similar, first element works row-wise, and the data directly before and after a given point is irrelevant; the second element works purely column-wise, focusing on inclination alone, where past datapoints carry all the relevant information.

To combine these two methods of prediction a custom neural network architecture is necessary, as there are no off-the-shelf algorithms suited for this task. This was achieved using Keras [81], developed and maintained by a Google engineer François Chollet. It works as a high-level input to TensorFlow [82], an ML library developed by Google Brain team. Using these tools it is possible to build a custom architecture using implemented building blocks such as vanilla RNN, LSTM, GRU, densely connected ANN layer (Dense), as well as additional non-learning layers that are useful for various reasons, such as dropout layers, gaussian noise, concatenation etc.

Figure 3.9 presents the final model architecture developed and used in this, or slightly modified form for multiple publications. Publication [AT3] introduced

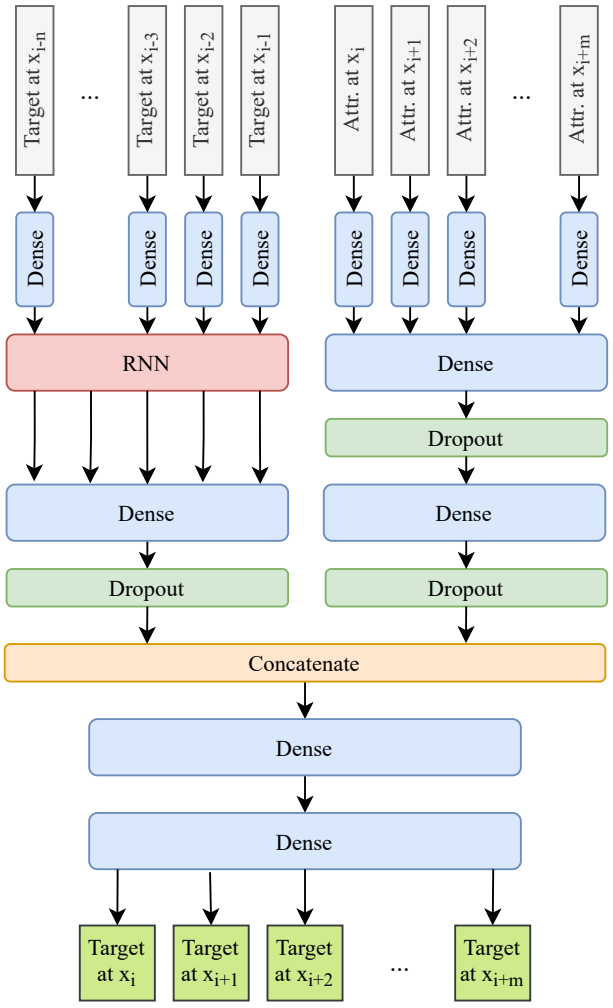


Figure 3.9: Developed branched model architecture [AT1]

the model, which was later used from continuous learning perspective in [AT4]. The model contains two distinct branches: one for the RNN input, and other the MLP input. The left hand side, RNN, will accept the inclination data from points from x_{i-n} to x_{i-1} (ref. Figure 3.8 for visual representation). The data is first channeled through dense layer for basic processing, and is later put into an RNN layer, which will be either LSTM or GRU; inclination case study showed better results with the GRU layer, while work focused on ROP prediction indicated better results with LSTM [AT8]. The result is put into a densely connected ANN

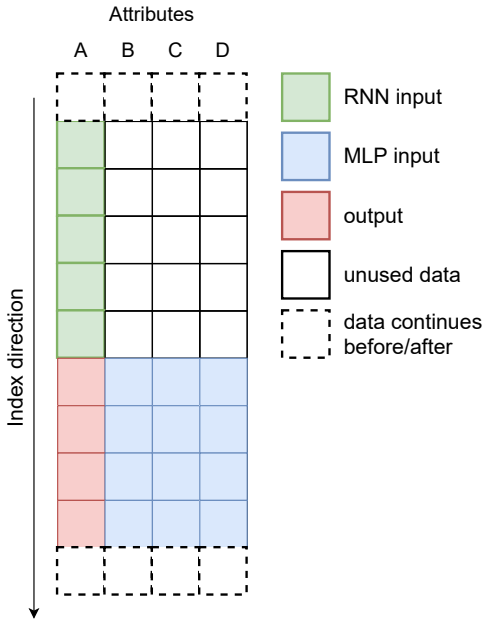


Figure 3.10: Sample shape for proposed branched neural network, first published in [AT1]

layer and a dropout layer afterwards that is randomly dropping connections while training to enable better generalization abilities of the network. This layer is set to drop a specified percentage of the connections, which is a hyperparameter to be tuned.

The other branch use real-time parameters as input from location x_i to x_{i+m}. First attributes are put through row-wise dense layers, later to act as an input to a sequence of dense-dropout-dense-dropout layers.

To further visualize the way the inputs are arranged Figure 3.10 is shown; this visualizes where the inputs are in relation to the output. In earliest publications on the proposed model [AT3, AT4] the area here marked as *unused data* was also used for MLP input, however it does not directly correspond to the output and by removing the prediction quality increased.

To visualize a sample with practical data Figure 3.11 is shown. It is a single sample from the case study, where Surface Torque, Rotary Speed and ROP

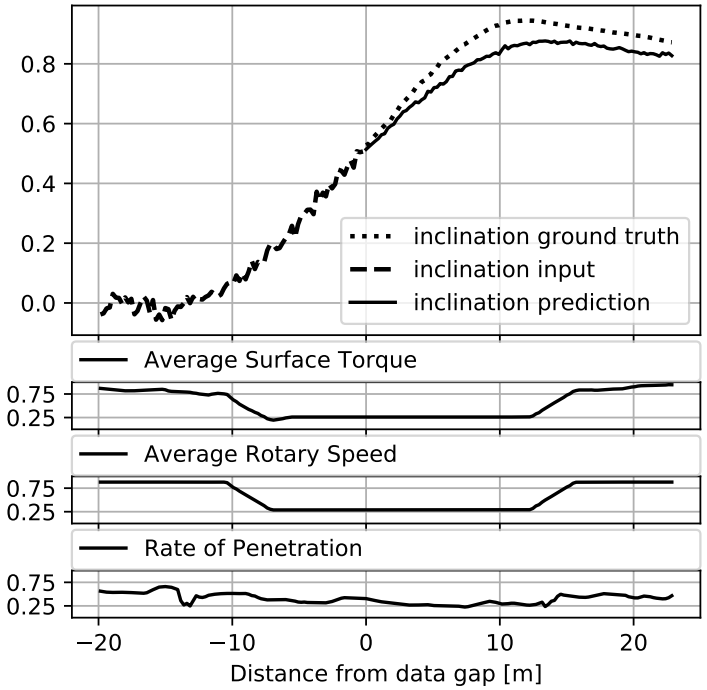


Figure 3.11: Input sample, originally published in [AT5]

are known in real time, while Inclination data is lagging behind and has to be predicted. The continuous line in the upper chart is the prediction done by the model and the dotted line is ground truth. All other values are inputs. In this particular example taken from earlier work [AT3, AT5] inputs were selected using expert judgement, and as indicated before contain MLP input throughout the whole length of the sample, ie. also cover known inclination region, not only the prediction.

Chapter 4

Sensitivity analysis

4.1 Rationale

Machine learning models are notorious for being so called *black boxes*; this means that it is not straight-forward to understand what exactly the model is doing. While one can display the structure of ANN with all the weights and biases established throughout the training process at a certain complexity level it becomes impossible to understand the logic behind an output to a given input manually. If a model is not well understood it brings a risk when it is to be deployed.

It is possible to gain some insight to the inner workings of a model by performing sensitivity analysis, and this chapter focuses on providing method of doing that for RNN-based models, particularly for the proposed branched model architecture described in this dissertation. Presented methods are flexible enough to be applied to other, significantly different architectures as well.

4.2 Definition

Sensitivity can mean different things, therefore specifying which definition is discussed here specifically. Sensitivity analysis in a sense discussed in this dissertation is defined as the ratio of the change in model's output relative to the change in model's input.

The reason for sensitivity analysis is the unknown nature of complex models, where relationship between inputs and the output is not clear. Such models are often referred to as 'black boxes'. Sensitivity analysis is typically performed together with uncertainty analysis, analyzing the range of potential outputs [83].

The basic approach to sensitivity analysis is to change one of the input parameters and to observe the change of the output [84]. This method is often referred to as one-at-a-time (OAT). Its biggest downside is that it is unable to efficiently explore the input hyperspace for highly dimensional models [85] making the method unreliable for them.

Different method for sensitivity analysis is *regression analysis* [86], where the input and output relationship is fitted with a linear regression and sensitivity is taken from the appropriate regression coefficients.

Variance-based sensitivity, often referred to as *Sobol' indices* [87, 88], is a more robust method of calculating sensitivity of a system. It decomposes the variance of the output to signify what portion of it is related to a given parameter. Furthermore, this method allows also to uncover that variance is dependent on interaction of two inputs. The calculation is based on sampling random points in the input domain; often a latin hypercube method [89] retaining the statistical probability of inputs is utilized to reduce the necessary amount of calculations as well as make the analysis more reliable. Additional noteworthy variance based method is Fourier amplitude sensitivity testing (FAST) [90]; it utilizes fourier series to represent the model and reduce the computational complexity compared to competing methods.

A relatively new method, called Variogram Analysis of Response Surfaces (VARS) [91] that aims to provide deeper understanding of sensitivity than the partial derivative and the variance based methods. This is achieved by classifying sensitivity not through a single value, but as a variogram [92], allowing to better define the sensitivity in the input hyperspace.

4.2.1 Sensitivity analysis in Machine Learning

Recent publication analyzed an ML model for predicting real estate prices [93] utilized a Profile method of sensitivity analysis [94] which in principle is an extension of partial derivative, where five alternative scales of input change are probed.

Different publication from 2013 focuses on proposing a number of novel methods of sensitivity analysis for data driven models [95]. Following methods were presented:

- 1D-SA - OAT PaD method
- GSA - an earlier published global sensitivity analysis [96]
- Data-based SA - PaD where few samples are taken as the starting points
- Monte-Carlo SA - PaD method where N starting points samples are generated synthetically
- Cluster-based SA - utilizing clusters of inputs based on the dataset

Another publication [97] focuses on exploring sensitivity analysis in ML vision systems, employing partial derivatives, creating heatmap of importance of individual pixels in a given image. In general, sensitivity analysis within Machine Learning is far from unexplored, but the publications on the topic are relatively sparse and there are no widely adopted standards in use.

4.2.2 Sensitivity analysis of ML in drilling

Sensitivity analysis is rarely performed in relation to ML models in drilling. Most commonly mentioned sensitivity analysis performed in relevant research refers to optimizing the ML model, or hyperparameter tuning [98, 99, 100], treating it as a meta-model where error is to be optimized. Only one publication was identified that strictly performed sensitivity analysis of the ROP via ML surrogate model in relation to the model inputs [101]. This analysis was performed via one-at-a-time method, changing one input while keeping others constant; single base case, here a mean value was used. More recently same author performed sensitivity analysis to explore modelled drill bit behaviour [102]. The publication produced results as cobweb plots, which are sometimes used to visualize sensitivity [103], which are well suited for exploring systems with relatively low quantity of variables, and ML models with recurrent elements may contain hundreds.

In the domain of artificial neural networks it is possible to evaluate sensitivity though analysis of the weights attached to neurons [104]; again an approach that is problematic in practical implementation for bigger networks.

4.3 Local, global, and data-driven sensitivity

Sensitivity analysis can be performed in a twofold way in terms of the scope of inputs. Basic approach considers sensitivity around a single set of inputs and is considered a **local sensitivity** analysis, or one-at-a-time analysis. This approach is simpler, but the results are valid either for only close vicinity of the input vector, or for the whole system if the system is fully linear.

Considering a basic linear system: $y = f(x) = 2x$ Sensitivity of this equation for any given input will result in the same result. Derivative of a linear equation is constant value, and therefore partial derivative will also be a constant value.

Non-linear systems will have different sensitivity depending on the selected starting point. Most basic non-linear system, $y = x^2$, will have negative sensitivity (output

decreases when input increases) whenever starting point of input x is in range of $(-\infty, \sim 0)$. The sensitivity will be positive when selected starting point x is in range $(\sim 0, \infty)$. Results of a local sensitivity analysis for a non-linear systems can be used as an approximation of a global sensitivity, if the non-linearity is small, for example equation $y = x^2$ will be nearly linear if a system described by it realistically exists only for a very small range.

Alternative approach is **global sensitivity** analysis. There are various methods to calculate such sensitivity that is independent of the input, such as employing Monte-Carlo analysis [105] or via Sobol' indices [87, 88].

Research presented in this dissertation and the related publication [AT5] aims to perform global sensitivity analysis through partial derivative (PaD) that is **data-driven**. The innovation in this work is to perform such analysis through utilization of the dataset (inputs only) at hand which both defines the realistic area of the input hyperspace as well as statistical density of such hyperspace; the result is presented as statistical distribution providing additional sensitivity information.

4.3.1 Partial Derivative

The process presented in publication [AT5] is rooted in a partial derivative. Let's assume a system R defined by a function $f(\cdot)$ with input S as:

$$R = f(S) \tag{4.1}$$

To apply PaD sensitivity analysis one of the inputs is varied by a given small amount and change of the function in relation to the extent of change is calculated:

$$SI_{RS} = \frac{[R(S_0 + \Delta S) - R(S_0 - \Delta S)]}{2\Delta S} \tag{4.2}$$

where S_0 is a selected *starting* value of input and ΔS is the small change to the input. This is a *one-at-a-time* (OAT) approach, where only one input is changed a any given moment. This is shown to be problematic, since OAT methods cannot successfully probe the high-dimensional Euclidean space [85], and ML models including RNN elements, such as the one presented in this dissertation, can have hundreds of inputs, and therefore hundreds of dimensions of the input vector.

The goal however is not to analyze sensitivity in each and every possible location of the input hyperspace, but only in the realistic and therefore relevant areas of it. When RNNs are applied it is specifically because the sequences of data, and therefore sequences of inputs, are not arbitrary, but follow specific patterns. In the case study of inclination prediction in directional drilling it is the smooth sequences of sliding and rotating of the bent sub creating characteristic wavy signal. Inclination will not vary in a sawtooth-like signal and therefore even though the system will have *a* sensitivity for this input, it is irrelevant.

Data driven methods have a unique characteristic which identify the area of realistic inputs - by definition the training dataset for an ML model has to cover a realistic range of inputs. If the dataset at hand does not cover realistic input range it will not work as intended in deployment.

4.4 Data driven sensitivity

Partial derivative based data driven sensitivity analysis is proposed [AT5], that significantly expands the previously published work on Data-based Sensitivity Analysis [95]; it is done on a larger scale and results are explored statistically. It performed by applying sensitivity shown in equation (4.2) to a given input S multiple times using the dataset at hand (used for model creation) as a set of starting points S_0. To formalize the process the dataset at hand is defined as as X; it can be containing both training and testing data, or just testing data for more stringent testing where generalization capabilities of the model at hand are not yet established. This dataset will contain n samples $[X_1, X_2, ..., X_n]$. Each

individual sample X_i, with i designating a given sample, will contain an input vector of length m:

$$X_i := [x_{i,1}, x_{i,2}, ..., x_{i,m}] \tag{4.3}$$

The data driven sensitivity index (DDSI) for a given input from 0 to m will be a set of PaD sensitivity indexes SI calculated via equation (4.2) $DDSI := [SI_1, SI_2, ..., SI_n]$. Note that any given SI will be a vector if the model returns a vector as well.

Application requires selection of the value Δ. Following best ML practices all inputs are typically scaled to range (0,1) or similar. In research presented in this dissertation we have used $\Delta = 0.1$, ie. 10% of input value change. Multiple values of Δ can be explored to evaluate sensitivity to a small or a big change.

Resultant DDSI is a three-dimensional vector with dimensions responsible for sample, input, and output. This allows to perform statistical analysis of the sensitivity. Example using inclination sensor lag case study discussed in detail in the attached publication [AT5]. An example taken from this study shows a 2D image, where the sample direction of the DDSI vector is reduced to an average, resulting in visualization of average sensitivity of a given input for a given output. The results shown in Figure 4.1 are truncated to show only inclination data input (data behind the directional sensor) relationship to output - the predicted data ahead of the sensor.

4.4.1 Sensitivity per input channel

Calculating sensitivity for an individual input however may not necessarily be the most informative way towards understanding how a given ML model behaves. The particular inclination prediction model that was evaluated consisted of 644 inputs (86 inclination inputs, 3x 186 associated attributes inputs) as well as 100

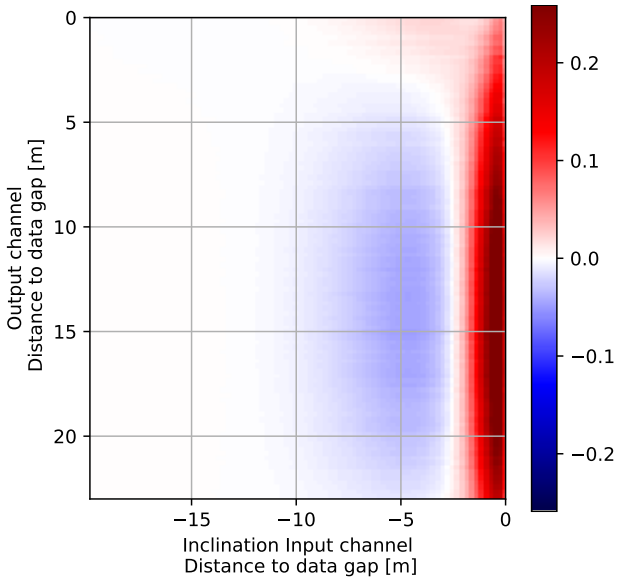


Figure 4.1: Sensitivity heatmap between Inclination input channel and output, with color bar designatig data driven sensitivity; from [AT5].

outputs for inclination prediction. These values are tied to the architecture of the problem at hand, available attributes, desired prediction distance, resolution, etc.

DDSI method presented in [AT5] can be applied towards complete *channel*, which here designates all the inputs related to a given attribute, for example inclination. The process is analogous to DDSI calculated per individual input. This way output response to change of all inputs out of a given channel is measured.

Calculating DDSI per channel provides new result visualization opportunities, with many variants presented in related publication [AT5]. One example is reproduced here in Figure 4.2, where sensitivity is evaluated to inclination input as a whole. It is a heatmap of results generated from multiple samples corresponding to multiple starting points. While the y-axis shows sensitivity, the x-axis designates prediction at various distances from the sensor. It effectively shows that the prediction close to the sensor is strictly tied to the absolute value of the inclination input, while further predictions are not as clear, although on average still the sensitivity is approximately equal to 1.

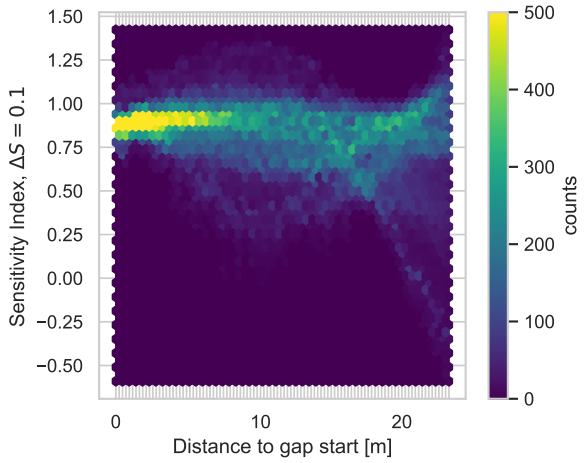


Figure 4.2: Statistical sensitivity to 10 percent value change on Inclination channel, heatmap. Here "gap start" is the location of the sensor, after which there si a data gap; from [AT5].

4.5 Comparison with Sobol' indicies

As discussed earlier in this chapter, a popular sensitivity analysis method is a variance-based global sensitivity analysis called Sobol' indices [87]. It calculates how much outputs' variance can be assigned to individual inputs. Compared to DDSI there are a number of limitations to this method. Firstly, using Sobol' indicies the sign of correlation is unknown (proportional or inversely proportional), requiring additional model exploration when such information is critical. Another key difference is that it is not possible to calculate Sobol' indices per complete channel, only individual inputs. With ML models for data-series taking inputs from different data sources considering offset for a complete signal line is an important consideration when analyzing sensitivity to a sensor drift or miscalibration.

Additional practical difference is that DDSI method can be used to calculate sensitivity of a very small portion of the model, for example a one given input, without needing to explore sensitivity of the all inputs, which is necessary in

Sobol' indicies method. As the ML models get more complex quick, meaningful results related to a small subset of inputs may have a big practical use.

Publication introducing the data driven sensitivity analysis [AT5] presents results of calculated Sobol' indicies as well as presents a thorough comparison between them and the newly proposed method. For a quick reference, Figure 4.3 is reproduced, which shows Sobol' indicies result for the analogous setup as shown in Figure 4.1. The information that the sign of the sensitivity changes in a portion of the input-output mapping is lost, and overall results have less definition to them. This is likely due to fact that Sobol' indicies are calculated without regard to how the realistic inputs are constructed, analyzing the ML model with random data.

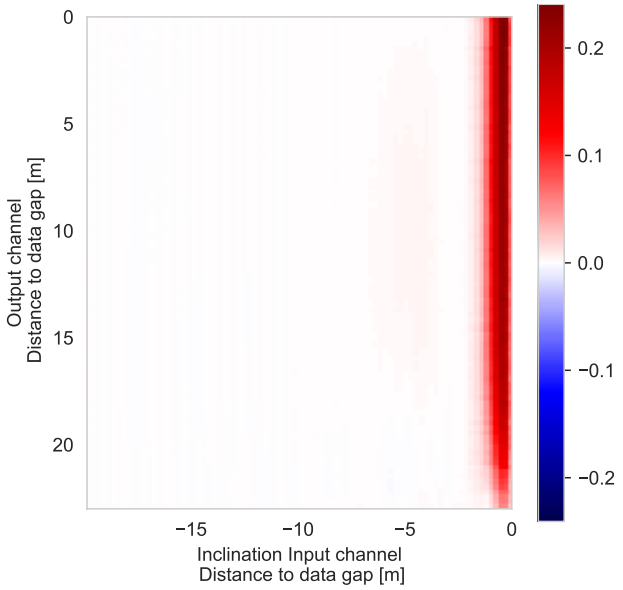


Figure 4.3: Sobol indices heatmap of Inclination input channel and output, colorbar indicates the value of Sobol' indicies; from [AT5].

Chapter 5

Contributions

5.1 Publications

Number of first author publications were published in relation to the PhD studies comprising this dissertation, reflecting research performed between August 2019 and late 2021. There are total five first-author publications; one is connected to Conference on Ocean, Offshore and Arctic Engineering (OMAE) in 2020, classified as level 1, and four published in the Journal of Petroleum Science and Engineering (Elsevier).

The publications together contribute to key elements of an applied ML process workflow from raw data, through data processing to model creation, training, and analysis, as presented in Figure 5.1. The publication numbers are provided here for the ease of reading, and do not reflect publication sequence.

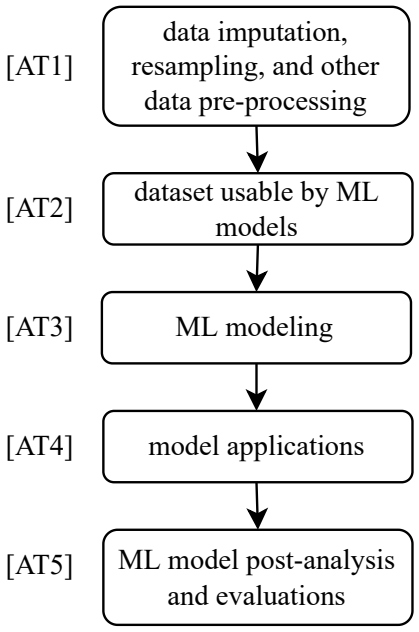


Figure 5.1: Mapping of contributions, listing relevant publications

5.1.1 Publication [AT1] - Impact of data pre-processing techniques on recurrent neural network performance in context of real-time drilling logs in an automated prediction framework

This publication focuses on the next step in the basic applied ML workflow, data preparation - here in context of drilling logs with focus on the RNN application. Recent literature review pointed out that this problem is mostly omitted [17], hence has a good research potential, especially for more advanced ML models. This publication answers a question about the impact of data preparation in context of RNN models in drilling. In this publication the Gap Coefficient concept is introduced, together with a complete process for implementing RNNs. Difference between forward filling and linear interpolation on results is explored, together with proposal of selection algorithm helping to choose the appropriate algorithm for each attribute. Resampling strategies, like KNN and RNR, are

compared and a novel method of evaluating resampling quality is proposed, the RMRS.

All the aspects related to data preparation can be considered novel in this publication. The topic of drilling data preparation is fairly poorly researched to date, in big part because in a lab setting it is easy to avoid the issues that have to be dealt with when working on real drilling data.

Key contributions to this publication are the conceptual and practical development of the pre-processing methods, development of the complete framework for data prediction using RNN, as well as benchmarking of different approaches presented. Key scientific finding of this publication is that pre-processing methods play a significant role in the accuracy of the model. There is a small, yet significant difference between imputation based on forward filling and linear interpolation. Explored case studies show that instead of input selection based on traditional methods, such as correlation coefficients, PCA can be used to reduce the dimensionality of the problem and in return yield lowest error.

5.1.2 Publication [AT2] - Reference dataset for rate of penetration benchmarking.

To create a tangible, usable product of the pre-processing efforts, as well as to use the developed methods for practical model application, a curated dataset for ROP prediction was created. Therefore, this publication has more of a goal than a research question - to establish best practices for machine learning application in drilling, and at the same time provides a benchmarking dataset that can be and is used by other researchers to evaluate different approaches to ROP prediction. Based off the Volve dataset seven wells were hand-picked with a number of common attributes, pre-processed and resampled. Three realistic scenarios were presented:

a) All for One - six wells are used for training, one for testing

b) One for All - one well is used for training, six for testing

c) Continuous learning - employing training-while-drilling scenario, where training data is continuously obtained and close future is predicted.

Additionally reference results are presented from a number of established ML algorithms to help others gauge any potential improvements over the baseline. Main innovation and the scientific contribution behind this publication is to provide a benchmark dataset for applied machine learning in drilling, something that did not exist to date. This is a very valuable resource already used by students and researchers alike, with some work already referencing the publication [106, 107]. Key contribution to this publication was hand picking all the data from the Volve dataset, creating the reference scenarios as well as calculating the base-case results using off-the-shelf ML algorithms.

5.1.3 Publication [AT3] - Continuous drilling sensor data reconstruction and prediction via recurrent neural networks

Continuing in the applied ML process shown in Figure 5.1, focusing on the model itself, this publication answers a question whether inclination data can be made known in directional drilling with bent sub for the region between the drill bit and the inclination sensor. During directional drilling with bent sub assembly the key sensors related to directional drilling - inclination, and azimuth - are located at a significant distance from the drill bit, in the MWD unit. Data from the Volve dataset suggests that this distance is often approximately 23 meters, which will differ depending on the technology used. As the targets that the directional driller has to hit are just few meters tall and wide, this sensor lag is a significant issue. While the steering ability of the BHA is limited, especially in the case of the bent sub, inclination and azimuth will change due to downhole conditions without explicit input from the driller. The position of the well is in principle not known for that region, except for an estimate by the driller.

To solve the sensor lag issue a custom branched RNN is introduced to predict the inclination data in the section of the well that is drilled, but it is not measured yet. Both temporal pattern in the inclination data as well as the correlation with real-time parameters that can be recorded instantly, such as torque, weight on bit, or rate of penetration, and is used for accurate prediction. The results show that the inclination data can be accurately predicted using the presented ML method with an average absolute error under 0.4°up to 20 meters ahead of the last known inclination value.

Main scientific contribution in this publication is introduction of the custom RNN network architecture, when RNNs were sparsely used before in research related to real-time drilling data. The key element to high accuracy was in the feature engineering, where inclination data was converted to a local coordinate system, a solution that was also compared to using inclination change, ie. the first numerical derivative. Additionally, the case study of delayed inclination sensor has a big practical use, as commercial actors strive to bring their sensors as close to the bit as possible. This solution provides reliable *inclination readings* at the bit, despite the sensor being 23 meters behind, using software only.

Work presented in this publication is later expanded in publication [AT4], where the model developed here is used in the continuous learning fashion. Furthermore, Publication [AT5] uses the model as a case study for the presented sensitivity analysis method.

5.1.4 Publication [AT4] - Training-while-drilling approach to inclination prediction in directional drilling utilizing recurrent neural networks.

Expanding on the previous publication, Publication [AT3], this publication takes the inclination prediction case study and focuses on the practical application of the model. It answers a question of the effects of practical application of an RNN model for inclination prediction, and whether it is feasible to train the model based on data received while drilling, without pre-training using historical data.

One of the typical problems in applied machine learning is the fact that the model has to be trained on data obtained from a system equivalent to the system that is meant to be later predicted. This is a big practical problem, since there are often minute differences between the BHA setup, well plan, lithology etc, even between offset wells, making application in drilling difficult. The presented solution is a concept called continuous learning, where a model is continuously trained on fresh data. Here a case study is presented, where training data is taken from a well being drilled, and the testing is performed on the near-future data, exploring a situation where an untrained model is deployed and it "learns on the job".

Key scientific contribution and novelty of this publication is in the application of high resolution continuous learning application, or training-while-drilling. Earlier publications [38, 36, 39] presented similar approach in few, hand picked steps, while this research presents a continuous spectrum of results along the drilled well. It shows that in some cases of continuous process, such as directional drilling, it is possible to forgo training a model based on reference data, and simply use the data at hand.

5.1.5 Publication [AT5] - Data-driven sensitivity analysis of complex machine learning models: A case study of directional drilling.

As the final stage, this work is focused on analyzing the developed ML model. The research question is how can the sensitivity of a black-box ML model can be evaluated, with sensitivity being understood per definitions stemming from control theory, ie. how much does the output change per given change in input. This publication employs partial derivative, a tried and true sensitivity analysis method and expands it through the use of a reference dataset as a source of starting points. The main premise of this research is to apply the partial derivative not at an arbitrary starting point, but on multiple points taken from the dataset used to create the model in the first place. This solves the main issue of the partial derivative, one-at-a-time method, where sensitivity is explored in just

one point of the hyperspace of possible inputs. Furthermore, the method is particularly suitable for RNNs, where multiple inputs correspond to the same parameter at different points in time or space. Instead of polling sensitivity of single input, multiple inputs can be tested at the same time as a single input channel, ie. multiple readings of an attribute in time. Sensitivity results are analyzed in this method as a statistical distribution of results. In this publication presented method is applied both on a basic flow equation, as well as the delayed inclination sensor case study with branched RNN network.

The innovation behind this publication is a novel sensitivity analysis method, that allows to peer into the typical black-box machine learning model using traditional methods.

5.2 Petroleum perspective, machine learning perspective

Being multidisciplinary work, this dissertation falls between two areas of science - petroleum and machine learning. One can however appreciate, that in principle petroleum is a domain, while machine learning is a method; this dissertation therefore focuses on a method within a domain. Petroleum studies are after all rooted in mechanical, control, chemical, and geophysics domains.

Presented work is interesting from the petroleum perspective, as this research covers holistically the complete workflow needed to apply machine learning in real-time drilling. Fit for purpose data pre-processing, model, model application and model analysis is explored, providing novel methods and solving issues that were extremely difficult when utilizing traditional approaches.

This dissertation is also valuable from machine learning, or rather applied machine learning perspective. Pre-processing methods developed are not by any means limited to the domain of petroleum. Presented branched model is novel and useful for other applications - for example predicting power usage in households,

where RNN element models day-night and weekday-weekend patterns, and at the same time data can be fed to the MLP branch from the weather forecast, combining these two data streams into a solution superior to either prediction methods separately.

Presented holistic approach focused on the complete ML process starting from raw-data and finishing at model analysis depending on perspective will fall under petroleum, or applied ML - not between two fields, but simply within one field depending on the readers' perspective.

5.3 Other published research

5.3.1 Drilling dataset exploration, processing and interpretation using Volve field data.

This publication [AT6] focuses on raw data itself and performs the analysis of the Volve dataset published in 2018 by Equinor. The research question behind this publication is to understand what data is available in the real-time drilling dataset. It presents the work performed to convert the WITSML files to CSV files and makes them available for others to use. The publication discusses various problems related to data quality. Examples are shown and multiple alternatives are presented in relation to filling in the data gaps. By describing exactly what kind of real-time drilling data is available, together with quantity of the wells, idiosyncrasies related to data being available in both time-based and depth-based formats, it acts as a catalyst for future research [108]. It contains information about typical data problems that one will encounter while using this data, together with description of the root cause, as well as best practices on how to deal with them.

The main scientific contribution in this publication is making the real time drilling data better understood and providing the easy to read CSV files for others to

use, as well as providing better understanding of the data available in the Volve dataset.

5.3.2 Automated Iterative Gap Filling Method for Drilling Logs and Other Data Series

This work [AT10] focuses on the data-preprocessing aspect of imputation. It presents an effort in creating a novel, iterative, sudoku-like, semi brute-force approach to gap filling, where gaps in all of the attributes are attempted to be filled by using various algorithms applied in various manner. A matrix of results is constructed and the best ones, that are within designated R^2 threshold, are applied. The algorithm is then repeated until all the gaps are filled. Synthetic as well as drilling-based case studies are presented that show a significant improvement over the other fully automated methods of gap filling.

The method presented in this publication is completely novel and not based off other work; it answers the question whether it is feasible to automate and at the same time improve the quality of data imputation over other fully automated methods. While the quality aspect was achieved in the evaluated benchmark, the feasibility remains inhibited by the excessive computational time required by the method.

5.3.3 Drilling data quality improvement and information extraction with case studies

This publication [AT7] focused on the drilling data quality, mainly on filtering, outlier removal, data assimilation, data validation and reconciliation, etc. Presented methods improve accuracy, consistency, reliability and validity of the available data, and the work is presented using both lab collected data, acquired from University of Stavanger Drillbotics team, as well as from the Volve real-time drilling data published by Equinor.

5.3.4 Downhole Data Correction for Data-driven Rate of Penetration Prediction Modeling

This publication [AT8] is utilizing an RNN architecture to predict ROP for different prediction horizons utilizing physics-based DWOB correction. The goal of this publication is to highlight the limitations of pure machine learning approaches and the benefits of physics-based feature engineering efforts that can vastly improve the results. The ML method with corrected DWOB is benchmarked against alternative approaches, as well as evaluated against different prediction horizons - predicting 5, 7.5, and 10 meters ahead of the bit.

Chapter 6

Conclusions

Employing RNN in the domain of drilling requires handling various idiosyncrasies of the drilling process. High quantity of asynchronous data and unique model deployment requirements necessitate holistic approach to the ML implementation process, with individual conclusions ranging from data preparation to model analysis.

1. Modern data-logging produces out-of-sync readings due to ever increasing quantity of sensors of various type. Necessary pre-processing techniques related to resampling such data have a significant impact on the quality of the results while using ML methods utilizing temporal information, such as RNN, LSTM, GRU [AT1]. To date this is a fairly unexplored topic in ML research related to drilling and can be considered an untapped source of additional, available performance improvement.

2. Resampling, which is necessary due to the way drilling data is recorded, can significantly deform the data. This can result in apparent increase in the performance of a given algorithm, while in reality additional significant error is added in the data preparation process. Such difference between two data-series can be efficiently calculated numerically as root of mean square distance despite the fact that are not sharing the same index values [AT1].

Quantification of data resampling performance is a significant element of understanding the data at hand.

3. Attribute selection is an important early step in developing a ML model and various techniques can be used to provide the best selection and reduce the error. PCA as dimensionality reduction is a good alternative approach for drilling data that was shown in some cases providing significantly better performance [AT1]. With drilling logs containing a lot of stationary and duplicated parameters, PCA transformation retains maximum of data variance per given quantity of attributes, making it a good solution for ML models in drilling.

4. There is a strong need to systematically compare ML methods applied in drilling and lack of common datasets hinders progress in the domain. Due to nature of the petroleum industry sharing data is difficult and rare, as companies are often barred from disclosing drilling logs due to legal reasons. Clear reference data, predefined train-test setup, and common error measurements are necessary to translate proposed methods to real-world performance [AT2]. Without reference data it is not possible to compare performance of competing algorithms.

5. Inclination data missing from the area between the drill bit and the MWD in a case of bent-sub drilling can be reliably predicted using ML methods utilizing architecture combining RNN and MLP elements. Average absolute error under 0.4 degrees was achieved for prediction distance of 20 meters [AT3]. Implementing this method can significantly improve the accuracy of existing and future directional drilling tools without hardware investment or adjustment.

6. ML models, such as the one explored inclination data prediction, can be reliably deployed untrained, where training data is collected while drilling. In the aforementioned case study the model started providing reliable results after 180 meters of relevant drilling [AT4]. This not only simplifies the field deployment significantly, but also removes the doubt about matching

the model with the BHA at hand. While this method does not provide results from the first meter drilled, the simplicity of the solution offsets this limitation

7. Classical sensitivity analysis methods, as understood in control theory, are ill-suited for ML models that contain high (50+) number of inputs. Using data-driven sensitivity analysis allows for reliable inspection of the model within the domain covered by the dataset [AT5]. This method provides results that are linked to the potential inputs, and it also allows for sensitivity analysis for specific ranges of inputs that correspond to real-life scenarios.

6.1 Future work

Opportunities exist to significantly expand on the presented work. RNNs are still relatively young and their strengths in the domain of drilling are yet to be fully explored. Furthermore, even newer competing algorithms, such as transformers, require further investigation into how they can be best applied in drilling.

1. Extra research should be done into understanding why there is a significant difference between forward filling and linear interpolation when imputing small gaps in data such as show in publication [AT1]. Today the presented results are statistically significant and the source of this effect is yet to be explored from either practical or the mathematical perspective. Bigger scale evaluation is also worth performing to better establish where the effect is biggest and how reliably this difference can be identified.

2. Today, Riemann Squared method presented in [AT1] can quantify which resampling algorithm generates data closest to the ground truth. More work is needed to develop derivative methods that are needed to quantify the exact amount of error added to a typical sample. Error may vary throughout the dataset, and identification of areas contributing the biggest amounts will be highly valuable.

3. Reference datasets beyond just ROP [AT2] should be made to promote reproducible ML research in drilling. Furthermore, the USROP dataset can be expanded by adding lithology information that exists in the Volve dataset, making in even more valuable for other researchers. Lack of research reproducibility is one of the major concerns among ML researchers [34] throughout various domains.

4. Systematic exploration of additional architectures combining RNN and MLP branches [AT3] should be performed, as such structures can be used in many different domains beyond petroleum, for example predicting household power use based on trend and forecasted weather. Software available today allows for relatively simple creation of ML architectures from existing algorithms and is an attractive opportunity to explore in the domain of applied AI.

5. Exploration of additional case studies in directional drilling is necessary to fully understand the potential impact of reliable inclination prediction of data ahead of the sensor. While bent-sub drilling case study was explored based on Volve data, benchmarking on additional datasets is necessary to fully gauge the performance of proposed method. Expanding case studies to rotary steerable systems can also be beneficial - while those systems have typically close to the bit already, it may be possible to optimize the construction of the tool if the sensor location is no longer a priority.

6. Additional case studies for training-while-drilling methodology [AT4] should be explored, to establish which kind of problems can and which cannot be tackled using this method. Training the ML model on the job relieves the burden of creating a pre-trained model, making deployment much easier. Furthermore, there are ML technologies that allow to use a pre-trained model that can be later updated during deployment, alleviating the limitation of deploying completely untrained model.

7. Data driven sensitivity [AT5] concept should be explored for other types of sensitivity analysis beyond PaD, yielding better understanding of the black

box models. There are multiple ways of analyzing sensitivity in a model, and a systematic exploration of available algorithms and their potential suitability for data-backed methods can produce useful results.

Co-authored references

[AT1] Andrzej T. Tunkiel, Dan Sui, and Tomasz Wiktorski. "Impact of Data Pre-Processing Techniques on Recurrent Neural Network Performance in Context of Real-Time Drilling Logs in an Automated Prediction Framework". In: *Journal of Petroleum Science and Engineering* (2021). DOI: 10.1016/j.petrol.2021.109760.

[AT2] Andrzej T Tunkiel, Dan Sui, and Tomasz Wiktorski. "Reference Dataset for Rate of Penetration Benchmarking". In: *Journal of Petroleum Science and Engineering* (2020). ISSN: 0920-4105. DOI: 10.1016/j.petrol.2020.108069.

[AT3] Andrzej T Tunkiel, Tomasz Wiktorski, and Dan Sui. "Continuous Drilling Sensor Data Reconstruction and Prediction via Recurrent Neural Networks". In: *ASME 2020 39th International Conference on Ocean, Offshore and Arctic Engineering*. 2020. DOI: 10.1115/OMAE2020-18154.

[AT4] Andrzej T Tunkiel, Dan Sui, and Tomasz Wiktorski. "Training-While-Drilling Approach to Inclination Prediction in Directional Drilling Utilizing Recurrent Neural Networks". In: *Journal of Petroleum Science and Engineering* 196 (2021).

[AT5] Andrzej T. Tunkiel, Dan Sui, and Tomasz Wiktorski. "Data-Driven Sensitivity Analysis of Complex Machine Learning Models: A Case Study of Directional Drilling". In: *Journal of Petroleum Science and Engineering* 195 (Dec. 2020), p. 107630. ISSN: 09204105. DOI: 10.1016/j.petrol.2020.107630.

[AT6] Andrzej T Tunkiel, Tomasz Wiktorski, and Dan Sui. "Drilling Dataset Exploration, Processing and Interpretation Using Volve Field Data". In: *ASME 2020 39th International Conference on Ocean, Offshore and Arctic Engineering* (2020). DOI: 10.1115/OMAE2020-18151.

[AT7] Suranga C H Geekiyanage, Andrzej Tunkiel, and Dan Sui. "Drilling Data Quality Improvement and Information Extraction with Case Studies". In: *Journal of Petroleum Exploration and Production Technology* (2020). ISSN: 2190-0566. DOI: 10.1007/s13202-020-01024-x.

[AT8] Mauro Encinas, Andrzej T. Tunkiel, and Dan Sui. "Downhole Data Correction for Data-driven Rate of Penetration Prediction Modeling". In: *Journal of Petroleum Science and Engineering* ().

[AT9] *Kunnskapskanalen – Forsker Grand Prix 2020 - Stavanger – NRK TV*. URL: https://tv.nrk.no/serie/kunnskapskanalen/2021/KOID75005220.

[AT10] Andrzej T. Tunkiel, Dan Sui, and Tomasz Wiktorski. "Automated Iterative Gap Filling Method for Drilling Logs and Other Data Series". In: ASME 2021 40th International Conference on Ocean, Offshore and Arctic Engineering. June 21, 2021. DOI: 10.1115/OMAE2021-61927.

References

[1] Jake VanderPlas. *Python Data Science Handbook*. O'Reilly Media, 2016, p. 541. ISBN: 978-1-4919-1205-8. URL: https://jakevdp.github.io/PythonDataScienceHandbook/.

[2] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. *Dota 2 with Large Scale Deep Reinforcement Learning*. 2019. arXiv: 1912.06680.

[3] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Representations by Back-Propagating Errors". In: *nature* 323.6088 (1986), pp. 533–536. ISSN: 00280836. DOI: 10.1038/323533a0.

[4] Colin Shearer. "The CRISP-DM Model: The New Blueprint for Data Mining". In: *Journal of data warehousing* 5.4 (2000), pp. 13–22.

[5] *Volve Field Data Village Download - Data 2008-2016 - Equinor.Com*. URL: https://www.equinor.com/no/what-we-do/digitalisation-in-our-dna/volve-field-data-village-download.html.

[6] Equinor. *Volve Field Data (CC BY-NC-SA 4.0)*. 2018. URL: https://www.equinor.com/en/news/14jun2018-disclosing-volve-data.html.

[7] Suranga C.H. Geekiyanage, Dan Sui, and Bernt S. Aadnoy. "Drilling Data Quality Management: Case Study with a Laboratory Scale Drilling Rig". In: *Proceedings of the International Conference on Offshore Mechanics and Arctic Engineering - OMAE*. Vol. 8. American Society of Mechanical Engineers, June 2018, pp. -. ISBN: 978-0-7918-5129-6. DOI: 10.1115/OMAE2018-77510.

[8] Dan Sui, Olha Sukhoboka, and Bernt Sigve Aadnøy. "Improvement of Wired Drill Pipe Data Quality via Data Validation and Reconciliation". In: *International Journal of Automation and Computing* 15.5 (Oct. 2018), pp. 625–636. ISSN: 17518520. DOI: 10.1007/s11633-017-1068-9. URL: http://link.springer.com/10.1007/s11633-017-1068-9.

[9] Pradeepkumar Ashok, Adrian Ambrus, Dawson Ramos, John Lutteringer, Michael Behounek, Yueheng Lisa Yang, Taylor Thetford, and Terrence Weaver. *A Step by Step Approach to Improving Data Quality in Drilling Operations: Field Trials in North America*. Vol. All Days. SPE Intelligent Energy International Conference and Exhibition. Sept. 2016. DOI: 10.2118/181076-MS. eprint: https://onepetro.org/SPEIE/proceedings-pdf/16IE/All-16IE/SPE-181076-MS/1412664/spe-181076-ms.pdf. URL: https://doi.org/10.2118/181076-MS.

[10] Steffen Moritz, Alexis Sardá, Thomas Bartz-Beielstein, Martin Zaefferer, and Jörg Stork. *Comparison of Different Methods for Univariate Time Series Imputation in r*. 2015. arXiv: 1510.03924 [stat.AP].

[11] Irfan Pratama, Adhistya Erna Permanasari, Igi Ardiyanto, and Rini Indrayani. "A Review of Missing Values Handling Methods on Time-Series Data". In: *2016 International Conference on Information Technology Systems and Innovation (ICITSI)*. 2016, pp. 1–6. DOI: 10.1109/ICITSI.2016.7858189.

[12] Fredric J. Harris. "Multirate Signal Processing for Communication Systems". In: *Multirate Signal Processing for Communication Systems*. 2021, pp. i–lvii.

[13] N. S. Altman. "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression". In: *American Statistician* 46.3 (1992), pp. 175–185. ISSN: 15372731. DOI: 10.1080/00031305.1992.10475879.

[14] Mariano Gasca and Thomas Sauer. "On the History of Multivariate Polynomial Interpolation". In: ed. by C Brezinski and L B T - Numerical Analysis: Historical Developments in the 20th Century Wuytack. Amsterdam: Elsevier, 2001, pp. 135–147. ISBN: 978-0-444-50617-7. DOI: 10.1016/B978-0-444-50617-7.50007-0. URL: http://www.sciencedirect.com/science/article/pii/B9780444506177500070.

[15] William S Cleveland. "Robust Locally Weighted Regression and Smoothing Scatterplots". In: *Journal of the American Statistical Association* 74.368 (1979), pp. 829–836. DOI: 10.1080/01621459.1979.10481038. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1979.10481038.

[16] Shachar Kaufman, Saharon Rosset, and Claudia Perlich. "Leakage in Data Mining: Formulation, Detection, and Avoidance". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, New York, USA: ACM Press, 2011, pp. 556–563. ISBN: 978-1-4503-0813-7. DOI: 10.1145/2020408.2020496. URL: http://dl.acm.org/citation.cfm?doid=2020408.2020496.

[17] Luís Felipe F.M. Barbosa, Andreas Nascimento, Mauro Hugo Mathias, and João Andrade de Carvalho. "Machine Learning Methods Applied to Drilling Rate of Penetration Prediction and Optimization - A Review". In: *Journal of Petroleum Science and Engineering* 183 (Dec. 2019), p. 106332. ISSN: 09204105. DOI: 10.1016/j.petrol.2019.106332.

[18] Robert Clear and Sam Berman. "Estimation of Linear Interpolation Error". In: *Journal of the Illuminating Engineering Society* (1989).

[19] Bing Liu. "Lifelong Machine Learning: A Paradigm for Continuous Learning". In: *Frontiers of Computer Science* 11.3 (2017), pp. 359–361. ISSN: 20952236. DOI: 10.1007/s11704-016-6903-6.

[20] N. S. Altman. "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression". In: *The American Statistician* 46.3 (Aug. 1992), pp. 175–185. ISSN: 0003-1305. DOI: 10.1080/00031305.1992.10475879. URL: http://www.tandfonline.com/doi/abs/10.1080/00031305.1992.10475879.

[21] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. "Scikit-Learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[22] Zhengping Che, Sanjay Purushotham, Guangyu Li, Bo Jiang, and Yan Liu. "Hierarchical Deep Generative Models for Multi-Rate Multivariate Time Series". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 784–793. URL: http://proceedings.mlr.press/v80/che18a.html.

[23] Satya Narayan Shukla and Benjamin M. Marlin. "Interpolation-Prediction Networks for Irregularly Sampled Time Series". In: *7th International Conference on Learning Representations, ICLR 2019* (Sept. 2019). URL: http://arxiv.org/abs/1909.07782.

[24] Yulia Rubanova, Ricky T Q Chen, and David Duvenaud. "Latent Ordinary Differential Equations for Irregularly-Sampled Time Series". In: (2019).

[25] Saurabh Tewari, Umakant Dhar Dwivedi, and Susham Biswas. "A Novel Application of Ensemble Methods with Data Resampling Techniques for Drill Bit Selection in the Oil and Gas Industry". In: *Energies* 14.2 (2021), p. 432.

[26] Christopher C Harvey and Steven J Schwartz. "Time Series Resampling Methods". In: *Analysis methods for multi-spacecraft data, edited by: Paschmann, G. and Daly, PW* (1998), p. 43.

[27] Melvin B Diaz and Kwang Yeom Kim. "Improving Rate of Penetration Prediction by Combining Data from an Adjacent Well in a Geothermal Project". In: *Renewable Energy* 155 (2020), pp. 1394–1400.

[28] Jon L Bentley. *Survey of Techniques for Fixed Radius near Neighbor Searching*. Stanford Linear Accelerator Center, Calif.(USA), 1975.

[29] *Sklearn.Neighbors.RadiusNeighborsRegressor — Scikit-Learn 0.21.3 Documentation*. URL: https://scikit-learn.org.

[30] James F Epperson. "On the Runge Example". In: *The American Mathematical Monthly* 94.4 (Apr. 1987), pp. 329–341. ISSN: 0002-9890. DOI: 10.1080/00029890.1987.12000642. URL: https://doi.org/10.1080/00029890.1987.12000642.

[31] Evelyn Fix. *Discriminatory Analysis: Nonparametric Discrimination, Consistency Properties*. Vol. 1. USAF school of Aviation Medicine, 1985.

[32] Tomasz Wiktorski and Aleksandra Królak. "Extended Approach to Sum of Absolute Differences Method for Improved Identification of Periods in Biomedical Time Series". In: *MethodsX* 7 (Jan. 2020), p. 101094. ISSN: 22150161. DOI: 10.1016/j.mex.2020.101094.

[33] Nicole Engelke and Vicki Sealey. "The Great Gorilla Jump: A Riemann Sum Investigation". In: *Proceedings of the 12th special interest group of the Mathematical Association of America on research in undergraduate mathematics education* (2009).

[34] Monya Baker and Dan Penny. "Is There a Reproducibility Crisis?" In: *Nature* 533.7604 (2016), pp. 452–454. ISSN: 14764687. DOI: 10.1038/533452A.

[35] Abdulmalek Ahmed, Abdulwahab Ali, Salaheldin Elkatatny, and Abdulazeez Abdulraheem. "New Artificial Neural Networks Model for Predicting Rate of Penetration in Deep Shale Formation". In: *Sustainability (Switzerland)* 11.22 (Nov. 2019), p. 6527. ISSN: 20711050. DOI: 10.3390/su11226527.

[36] Chiranth Hegde and K. E. Gray. "Use of Machine Learning and Data Analytics to Increase Drilling Efficiency for Nearby Wells". In: *Journal of Natural Gas Science and Engineering* 40 (2017), pp. 327–335. ISSN: 18755100. DOI: 10.1016/j.jngse.2017.02.019.

[37] Chiranth Hegde, Scott Wallace, and Ken Gray. "Using Trees, Bagging, and Random Forests to Predict Rate of Penetration during Drilling". In: *Society of Petroleum Engineers - SPE Middle East Intelligent Oil and Gas Conference and Exhibition* (2015). DOI: 10.2118/176792-ms.

[38] Chiranth Hegde, Hugh Daigle, Harry Millwater, and Ken Gray. "Analysis of Rate of Penetration (ROP) Prediction in Drilling Using Physics-Based and Data-Driven Models". In: *Journal of Petroleum Science and Engineering* 159 (Nov. 2017), pp. 295–306. ISSN: 09204105. DOI: 10.1016/j.petrol.2017.09.020.

[39] Cesar Soares and Kenneth Gray. "Real-Time Predictive Capabilities of Analytical and Machine Learning Rate of Penetration (ROP) Models". In: *Journal of Petroleum Science and Engineering* 172 (Jan. 2019), pp. 934–959. ISSN: 09204105. DOI: 10.1016/j.petrol.2018.08.083.

[40] Chiranth Hegde and Ken Gray. "Evaluation of Coupled Machine Learning Models for Drilling Optimization". In: *Journal of Natural Gas Science and Engineering* 56 (Aug. 2018), pp. 397–407. ISSN: 18755100. DOI: 10.1016/j.jngse.2018.06.006.

[41] Mohammad Sabah, Mohsen Talebkeikhah, David A. Wood, Rasool Khosravanian, Mohammad Anemangely, and Alireza Younesi. "A Machine Learning Approach to Predict Drilling Rate Using Petrophysical and Mud Logging Data". In: *Earth Science Informatics* 12.3 (Sept. 2019), pp. 319–339. ISSN: 18650481. DOI: 10.1007/s12145-019-00381-4.

[42] Jiahang Han, Yanji Sun, and Shaoning Zhang. "A Data Driven Approach of ROP Prediction and Drilling Performance Estimation". In: *International Petroleum Technology Conference 2019, IPTC 2019* (2019). DOI: 10.2523/iptc-19430-ms.

[43] Xian Shi, Gang Liu, Xiaoling Gong, Jialin Zhang, Jian Wang, and Hongning Zhang. "An Efficient Approach for Real-Time Prediction of Rate of Penetration in Offshore Drilling". In: *Mathematical Problems in Engineering* 2016 (2016). Ed. by Cheng-Tang Wu, p. 3575380. ISSN: 1024-123X. DOI: 10.1155/2016/3575380.

[44] B. Mantha and R. Samuel. "ROP Optimization Using Artificial Intelligence Techniques with Statistical Regression Coupling". In: *Proceedings - SPE Annual Technical Conference and Exhibition*. Vol. 2016-Janua. Society of Petroleum Engineers (SPE), Sept. 2016. ISBN: 978-1-61399-463-4. DOI: 10.2118/181382-ms.

[45] Tuna Eren and Mehmet Evren Ozbayoglu. "Real Time Optimization of Drilling Parameters during Drilling Operations". In: *SPE Oil and Gas India Conference and Exhibition*. Society of Petroleum Engineers, Apr. 2010. DOI: 10.2118/129126-MS.

[46] Cesar Soares, Hugh Daigle, and Ken Gray. "Evaluation of PDC Bit ROP Models and the Effect of Rock Strength on Model Coefficients". In: *Journal of Natural Gas Science and Engineering* 34 (Aug. 2016), pp. 1225–1236. ISSN: 18755100. DOI: 10.1016/j.jngse.2016.08.012.

[47] Omogbolahan S. Ahmed, Ahmed A. Adeniran, and Ariffin Samsuri. "Computational Intelligence Based Prediction of Drilling Rate of Penetration: A Comparative Study". In: *Journal of Petroleum Science and Engineering* 172 (Jan. 2019), pp. 1–12. ISSN: 09204105. DOI: 10.1016/j.petrol.2018.09.027.

[48] Khoukhi Amar and Alarfaj Ibrahim. "Rate of Penetration Prediction and Optimization Using Advances in Artificial Neural Networks, a Comparative Study". In: *IJCCI 2012 - Proceedings of the 4th International Joint Conference on Computational Intelligence*. 2012, pp. 647–652. ISBN: 978-989-8565-33-4. DOI: 10.5220/0004172506470652.

[49] Tuna Eren and Evren Ozbayoglu. "Real-Time Drilling Rate of Penetration Performance Monitoring". In: *Offshore Mediterranean Conference*

and Exhibition, 23-25 March, Ravenna, Italy. Offshore Mediterranean Conference, 2011. DOI: OMC-2011-076.

[50] A. T. Bourgoyne and F. S. Young. "A Multiple Regression Approach to Optimal Drilling and Abnormal Pressure Detection". In: *SPE Reprint Series* 14.49 (Aug. 1999), pp. 27–36. ISSN: 08910901. DOI: 10.2118/4238-pa.

[51] Ping Yi, Aniket Kumar, and Robello Samuel. "Real-Time Rate of Penetration Optimization Using the Shuffled Frog Leaping Algorithm (SFLA)". In: *Society of Petroleum Engineers - SPE Intelligent Energy International 2014*. Society of Petroleum Engineers (SPE), Apr. 2014, pp. 116–125. ISBN: 978-1-63266-413-6. DOI: 10.2118/167824-ms.

[52] Wanyi Jiang and Robello Samuel. "Optimization of Rate of Penetration in a Convoluted Drilling Framework Using Ant Colony Optimization". In: *SPE/IADC Drilling Conference, Proceedings*. Vol. 2016-Janua. Society of Petroleum Engineers (SPE), Mar. 2016. ISBN: 978-1-61399-401-6. DOI: 10.2118/178847-ms.

[53] M.G. Bingham. *A New Approach to Interpreting Rock Drillability*. The Petroleum Publishing Co., 1964.

[54] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Vol. 112. Springer, 2013.

[55] Jimin Tan, Jianan Yang, Sai Wu, Gang Chen, and Jake Zhao. *A Critical Look at the Current Train/Test Split in Machine Learning*. 2021. arXiv: 2106.04525.

[56] Kyle Gorman and Steven Bedrick. "We Need to Talk about Standard Splits". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 2786–2791.

[57] Rich Caruana, Steve Lawrence, and Lee Giles. "Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping". In: *Advances in neural information processing systems* (2001), pp. 402–408.

[58] C Reid Turner, Alfonso Fuggetta, Luigi Lavazza, and Alexander L Wolf. "A Conceptual Basis for Feature Engineering". In: *Journal of Systems and Software* 49.1 (1999), pp. 3–15.

[59] Jie Cao, Jiaxuan Gao, Tong Jiao, Ruiyi Xiang, Yanyan Pang, and Tiantai Li. "Feature Investigation on the ROP Machine Learning Model Using Realtime Drilling Data". In: *Journal of Physics: Conference Series*. Vol. 2024. 1. IOP Publishing. 2021, p. 012040.

[60] Shay Geller. *Normalization vs Standardization — Quantitative Analysis | by Shay Geller | towards Data Science*. 2019. URL: `https://towardsdatascience.com/normalization-vs-standardization-quantitative-analysis-a91e8a79cebf`.

[61] Jason Brownlee. *How to Choose a Feature Selection Method For Machine Learning*. Machine Learning Mastery. Nov. 27, 2019. URL: `https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/` (visited on 11/29/2021).

[62] Ke Yan and David Zhang. "Feature Selection and Analysis on Correlated Gas Sensor Data with Recursive Feature Elimination". In: *Sensors and Actuators B: Chemical* 212 (2015), pp. 353–363.

[63] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. "Pearson Correlation Coefficient". In: *Noise Reduction in Speech Processing*. Springer, 2009, pp. 1–4.

[64] Joost CF de Winter, Samuel D Gosling, and Jeff Potter. "Comparing the Pearson and Spearman Correlation Coefficients across Distributions and Sample Sizes: A Tutorial Using Simulations and Empirical Data." In: *Psychological methods* 21.3 (2016), p. 273.

[65] Florian Wetschoreck, Tobias Krabel, and Surya Krishnamurthy. "8080labs/Ppscore: Zenodo Release". In: (Oct. 2020). DOI: `10.5281/ZENODO.4091345`. URL: `https://zenodo.org/record/4091345`.

[66] Karl Pearson. "LIII. On Lines and Planes of Closest Fit to Systems of Points in Space". In: *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2.11 (1901), pp. 559–572.

[67] Mirac Suzgun, Yonatan Belinkov, and Stuart M Shieber. *On Evaluating the Generalization of LSTM Models in Formal Languages*. 2018. arXiv: 1811.01001.

[68] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems*. 2014, pp. 3104–3112.

[69] Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. "LSTM: A Search Space Odyssey". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (2017), pp. 2222–2232. DOI: 10.1109/TNNLS.2016.2582924.

[70] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. "A Comparison of ARIMA and LSTM in Forecasting Time Series". In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2018, pp. 1394–1401.

[71] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation". In: *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference* (2014), pp. 1724–1734. DOI: 10.3115/v1/d14-1179.

[72] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3 (2014). URL: http://arxiv.org/abs/1412.3555.

[73] Nicole Gruber and Alfred Jockisch. "Are GRU Cells More Specific and LSTM Cells More Sensitive in Motive Classification of Text?" In: *Frontiers in Artificial Intelligence* 3 (June 2020), p. 40. DOI: 10.3389/frai.2020.00040. URL: www.frontiersin.org.

[74] Qishuai Yin, Jin Yang, Mayank Tyagi, Xu Zhou, Ning Wang, Gang Tong, Renjun Xie, Hexing Liu, and Bohan Cao. "Downhole Quantitative Evaluation of Gas Kick during Deepwater Drilling with Deep Learning Using Pilot-Scale Rig Data". In: *Journal of Petroleum Science and Engineering* 208 (2022), p. 109136.

[75] Huan Xu, Chong Zhang, Geok Soon Hong, Junhong Zhou, Jihoon Hong, and Keng Soon Woon. "Gated Recurrent Units Based Neural Network for Tool Condition Monitoring". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2018, pp. 1–7.

[76] Augustine Uhunoma Osarogiagbon, Olalere Oloruntobi, Faisal Khan, Ramachandran Venkatesan, and Stephen Butt. "Gamma Ray Log Generation from Drilling Parameters Using Deep Learning". In: *Journal of Petroleum Science and Engineering* 195 (2020), p. 107906.

[77] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 08997667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735.

[78] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Vol. 13-17-Augu. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785.

[79] Caineng Zou. *Unconventional Petroleum Geology*. Elsevier, 2017, p. 64. ISBN: 978-0-12-397162-3.

[80] Michael I Posner and Marcus E Raichle. "The Neuroimaging of Human Brain Function". In: *Proceedings of the National Academy of Sciences* 95.3 (1998), pp. 763–764.

[81] François Chollet et al. *Keras*. \url{https://keras.io}. 2015.

[82] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. "Tensorflow: A System for Large-Scale Machine Learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016, pp. 265–283.

[83] Iain Macdonald and Paul Strachan. "Practical Application of Uncertainty Analysis". In: *Energy and Buildings* 33.3 (2001), pp. 219–227.

[84] David M Hamby. "A Review of Techniques for Parameter Sensitivity Analysis of Environmental Models". In: *Environmental monitoring and assessment* 32.2 (1994), pp. 135–154.

[85] Andrea Saltelli and Paola Annoni. "How to Avoid a Perfunctory Sensitivity Analysis". In: *Environmental Modelling and Software* 25.12 (2010), pp. 1508–1517. ISSN: 13648152. DOI: 10.1016/j.envsoft.2010.04.012. URL: http://dx.doi.org/10.1016/j.envsoft.2010.04.012.

[86] Jack PC Kleijnen. "Sensitivity Analysis and Optimization of System Dynamics Models: Regression Analysis and Statistical Design of Experiments". In: *System Dynamics Review* 11.4 (1995), pp. 275–288.

[87] Ilya M Sobol. "Sensitivity Estimates for Nonlinear Mathematical Models". In: *Mathematical modelling and computational experiments* 1.4 (1993), pp. 407–414.

[88] I. M. Sobol. "Global Sensitivity Indices for Nonlinear Mathematical Models and Their Monte Carlo Estimates". In: *Mathematics and Computers in Simulation* 55.1-3 (2001), pp. 271–280. ISSN: 03784754. DOI: 10.1016/S0378-4754(00)00270-6.

[89] Michael D McKay, Richard J Beckman, and William J Conover. "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code". In: *Technometrics : a journal of statistics for the physical, chemical, and engineering sciences* 42.1 (2000), pp. 55–61.

[90] RI Cukier, CM Fortuin, Kurt E Shuler, AG Petschek, and J Ho Schaibly. "Study of the Sensitivity of Coupled Reaction Systems to Uncertainties in Rate Coefficients. I Theory". In: *The Journal of chemical physics* 59.8 (1973), pp. 3873–3878.

[91] Saman Razavi and Hoshin Vijai Gupta. "Variogram Analysis of Response Surfaces (VARS): A New Framework for Global Sensitivity Analysis of Earth and Environmental Systems Models". In: *AGU Fall Meeting Abstracts*. Vol. 2015. 2015, H11D–1373.

[92] Emmanuel Gringarten and Clayton V Deutsch. "Teacher's Aide Variogram Interpretation and Modeling". In: *Mathematical Geology* 33.4 (2001), pp. 507–534.

[93] Thomas Dimopoulos and Nikolaos Bakas. "Sensitivity Analysis of Machine Learning Models for the Mass Appraisal of Real Estate. Case Study of Residential Units in Nicosia, Cyprus". In: *Remote Sensing* 11.24 (2019). ISSN: 2072-4292. DOI: 10.3390/rs11243047. URL: https://www.mdpi.com/2072-4292/11/24/3047.

[94] Muriel Gevrey, Ioannis Dimopoulos, and Sovan Lek. "Review and Comparison of Methods to Study the Contribution of Variables in Artificial Neural Network Models". In: *Ecological modelling* 160.3 (2003), pp. 249–264.

[95] Paulo Cortez and Mark J Embrechts. "Using Sensitivity Analysis and Visualization Techniques to Open Black Box Data Mining Models". In: *Information Sciences* 225 (2013), pp. 1–17.

[96] Paulo Cortez and Mark J. Embrechts. "Opening Black Box Data Mining Models Using Sensitivity Analysis". In: *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*. 2011, pp. 341–348. DOI: 10.1109/CIDM.2011.5949423.

[97] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. *Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models*. 2017. arXiv: 1708.08296.

[98] Mohammad Sabah, Mohsen Talebkeikhah, David A Wood, Rasool Khosravanian, Mohammad Anemangely, and Alireza Younesi. "A Machine Learning Approach to Predict Drilling Rate Using Petrophysical and Mud Logging Data". In: *Earth Science Informatics* 12.3 (2019), pp. 319–339.

[99] Osama Siddig, Hany Gamal, Salaheldin Elkatatny, and Abdulazeez Abdulraheem. "Real-Time Prediction of Poisson's Ratio from Drilling Parameters Using Machine Learning Tools". In: *Scientific Reports* 11.1 (2021), pp. 1–13.

[100] Ishank Gupta, Ngoc Tran, Deepak Devegowda, Vikram Jayaram, Chandra Rai, Carl Sondergeld, and Hamidreza Karami. "Looking Ahead of the Bit Using Surface Drilling and Petrophysical Data: Machine-learning-based Real-Time Geosteering in Volve Field". In: *SPE Journal* 25.02 (2020), pp. 990–1006.

[101] Peter Batruny, Hanif Yahya, Norazan Kadir, Amir Omar, Zahid Zakaria, Saravanan Batamale, and Noreffendy Jayah. "Drilling in the Digital Age: An Aproach to Optimizing ROP Using Machine Learning". In: *Abu Dhabi International Petroleum Exhibition & Conference*. OnePetro. 2019.

[102] Peter Batruny, Hafiz Zubir, Pete Slagel, Hanif Yahya, Zahid Zakaria, and Alaa Ahmad. "Drilling in the Digital Age: Machine Learning Assisted Bit Selection and Optimization". In: International Petroleum Technology Conference. Mar. 23, 2021, D012S045R137. DOI: 10.2523/IPTC-21299-MS. URL: https://doi.org/10.2523/IPTC-21299-MS (visited on 05/10/2022).

[103] Dorota Kurowicka and Roger Cooke. "Uncertainty Analysis with High Dimensional Dependence Modelling". In: *Uncertainty Analysis with High Dimensional Dependence Modelling. Wiley Series in Probability and Statistics*. Jan. 2006. ISBN: 0-470-86306-4. DOI: 10.1002/0470863072.

[104] Alsakran Jamal, Rodan Ali, Alhindawi Nouh, and Faris Hossam. "Visualization Analysis of Feed Forward Neural Network Input Contribution". In: *Scientific Research and Essays* 9.14 (July 2014), pp. 645–651. ISSN: 1992-2248. DOI: 10.5897/SRE2014.5895. URL: http://academicjournals.org/journal/SRE/article-abstract/F21242846447.

[105] Russel E. Caflisch. "Monte Carlo and Quasi-Monte Carlo Methods". In: *Acta Numerica* 7 (1998), pp. 1–49. ISSN: 14740508. DOI: 10.1017/S0962492900002804.

[106] Renan Gonzalo Ruiz Beviglia. "REAL-TIME DATA DRIVEN ROP AND TORQUE MODELLING AND OPTIMIZATION USING MACHINE LEARNING". uis, 2021.

[107] Seoyoon Kwon, Minsoo Ji, Gayoung Park, Baehyun Min, and Hoonyoung Jeong. "Analysis of Data Disclosure and Reservoir Model of the Volve Oilfield in the North Sea". In: (2021).

[108] Felix James Pacis. "An End-to-End Machine Learning Project for Detection of Stuck Pipe Symptoms during Tripping Operations". uis, 2021.

Appendix A

Appendix, publications

Attached publications related to this dissertation are reproduced below. They have been reformatted to fit the dissertation format, however no changes were done to the text, references, figures, tables, or equations. Reference numbering is restarted for each publication and a separate bibliography is produced at the end of each publication.

Appendix B

Impact of data pre-processing techniques on recurrent neural network performance in context of real-time drilling logs in an automated prediction framework, AT1

Tunkiel, A. T., Sui, D., & Wiktorski, T. (2021). **Impact of data pre-processing techniques on recurrent neural network performance in context of real-time drilling logs in an automated prediction framework** - Journal of Petroleum Science and Engineering 2021.

Abstract

Recurrent neural networks (RNN), which are able to capture temporal natures of a signal, are becoming more common in machine learning applied to petroleum engineering, particularly drilling. With this technology come requirements and caveats related to the input data that play a significant role on resultant models. This paper explores how data pre-processing and attribute selection techniques affect the RNN models' performance. Re-sampling and down-sampling methods are compared; imputation strategies,a problem generally omitted in published research, are explored and a method to select either last observation carried forward or linear interpolation is introduced and explored in terms of model accuracy. Case studies are performed on real-time drilling logs from the open Volve dataset published by Equinor. For a realistic evaluation, a semi-automated process is proposed for data preparation and model training and evaluation which employs a continuous learning approach for machine learning model updating, where the training dataset is being built continuously while the well is being made. This allows for accurate benchmarking of data pre-processing methods. Included is a previously developed and updated branched custom neural network architecture that includes both recurrent elements as well as row-wise regression elements. Source code for the implementation is published on GitHub.

B.1 Introduction

Nomenclature

ΔX	dataset, first numerical derivative
δ_{cf}	cutoff value
$\dot{x}$	first numerical derivative
$\hat{x}_{t+1}$	predicted value of x at t+1
$\theta_{th}\%$	threshold of zero-value numerical first derivatives
b	length of prediction (meters)
$g(\cdot)$	function describing difference between raw data and resampled data
GC_i	gap coefficient for gap length of i
h_t	cell output at current time t
j	number of attributes used
m	number of prediction (rows)
MSE_g	Riemann sum squared of function $g(\cdot)$
MSE_g	Root Mean Riemann sum squared of function $g(\cdot)$
n	number of memory (rows)
n_0	number of zero-valued first numerical derivatives

p	length of memory (meters)
$r(\cdot)$	function describing raw data
R^2	R-squared
S_g	Riemann sum of function $g(\cdot)$
SS_g	Riemann sum squared of function $g(\cdot)$
t	current sample time (row)
$t(\cdot)$	function describing resampled data
v_t	information from recurrent connection at time t
$x\%$	percentage of well drilled
x_t	input at current time t
CNN	convolutional neural network
CSV	comma separated values
d	distance in KNN/FRN
F	quantity of the well to be evaluated
F9A	One of the wells of the Volve dataset
FRN	fixed radius neighbour
GC	gap coefficient
GRU	gated recurrent unit
K	quantity of neighbours in KNN
k	quantity of datapoints of the first numerical derivative dataset
KNN	K-nearest neighbours
LSTM	long short-term memory

MAE mean absolute error

MD measured depth

ML machine learning

MLP multi-layer perceptron

MWD measurement while drilling

NLP natural language processing

PCA principal component analysis

r radius in FRN

RNN recurrent neural network

RNR Radius Neihgbour Regressor (implementation of FRN)

ROP rate of penetration

RPM revolutions per minute (rotary speed)

SPP stand pipe pressure

TL total length

u quantity of steps in Riemann sum

w weight in KNN/FRN

WOB weight on bit

B.1.1 Background and State of Art

Data preparation is often left as an afterthought when discussing machine learning (ML) research applied to drilling. Recently conducted review of rate of penetration (ROP) prediction papers [1] acknowledges the issue of data gaps in drilling logs; quoting directly from the aforementioned review paper: *In general, this problem*

was omitted. This is likely due to researchers working on datasets that are already pre-processed, where they are not exposed to this common practical problem. Raw drilling logs extracted from Volve field, dataset made public by Equinor [2], expose that they can be occupied in over 80% by data gaps [3]. Drilling logs from this dataset are used as a case study throughout this paper.

To capture the temporal behaviour of a given logged attribute recurrent neural networks (RNN) are commonly used [4]. Basic architecture of such network is shown in Figure B.1; cell A takes current input, x_t, as well as the information from recurrent connection v_{t-1} from the previous step; this may be simply the cell output h_{t-1}, but can also contain additional state information. This generates an output h_t. It is in practice implemented in so called unfolded state, as seen on the right hand side of the Figure B.1. Various designs of a RNN cell exist, with Long Short-Term Memory (LSTM) [5] and Gated Recurrent Units (GRU) [6] seeing wide implementations.

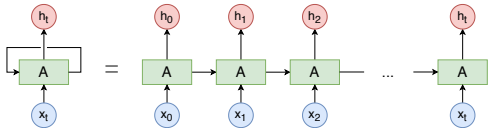


Figure B.1: Recurrent Neural Network basic architecture

When applying the RNN architecture one of the important assumptions is the equidistance of the samples along the indexed dimension. This means that samples are logged every x meters drilled, or every x seconds, where the value of x does not change throughout the dataset. This is rarely the case outside of the lab environment, and since the quantity of sensors used is ever increasing, this problem becomes more and more prevalent in the industry. To bring drilling log attributes in-sync and at constant sampling rate, data resampling step is necessary, where a dataset with uneven sampling rate is converted to a dataset that has a constant sampling rate. While reviewing the recent literature this step is often simply mentioned without even indicating which algorithm or the parameters are used [7]. To the best of authors' knowledge, petroleum related papers on the RNN rarely mention resampling [8, 9]; our previous work on the RNN [10] admittedly also sparsely touched the topic by simply mentioning

the algorithm used. It is difficult to find valuable, practical information and comments on the impact of resampling on RNNs also outside of petroleum, with some papers acknowledge the problem and mention "window based resampling" [11] without additional details. What further makes the literature review difficult is the fact that within ML the term *resampling* is also used for a method of removing the imbalance from the dataset, while for data-series, like in the case of real time drilling logs, resampling means creating a new dataset with a new, constant polling rate.

This paper focuses not only on the data preprocessing strategies and the effects on the quality of the models' results, but also on the attribute selection. This topic is widely discussed in most, if not all papers on topics like data- driven ROP modelling. In this paper we compared attribute selection strategies, applied dynamically through simulated drilling process, to evaluate which strategy is likely to yield better results. This is especially important when a given model is applied in a continuous learning approach, when the training dataset continuously expands and therefore correlation scores can dynamically change as the drilling progresses.

B.1.2 Motivation and Contribution

Some aspects have to be considered during ML models/applications development, like data quality issues, ML models' process automation and optimization, ML models' evaluation and interpretation etc. Motivation behind this paper is two-fold. First and foremost, closing knowledge gaps related to data pre-processing is identified as a a topic with an urgent need for attention. Capturing knowledge on temporal patterns becomes more common in the future, hence appropriate data preparation for this kind of problems is a crucial building block. The main contributions are in the following areas:

1. Method for automatic imputation algorithm is presented and evaluated for data gaps common in drilling data. (Data Quality Improvement)

2. Resampling process is heavily analyzed. First, evaluation in terms of the quality of resampling is done; and a novel method is introduced using Riemann sum to quantify how close the resampled data is to original data. Secondly, a case study is presented to show practical model performance improvements. (Data Quality Improvement)

3. The resampling rate is evaluated to gauge how reduced data resolution affects the machine learning model performance in a case study.(ML Model Evaluation)

This paper does not elaborate on the signal processing methods, such a median filter, Kalman filter, outlier identification etc. While this is an important step in data driven methods it is out of scope of this paper.

Secondly, to explore the effects of data pre-processing we have developed a semi-automatic framework for data preparation, modelling including temporal information, and evaluation in relation to problems from the realm of real-time drilling logs. It is challenging to provide universal answers on topics like attribute selection strategies in drilling, therefore providing tools to perform such studies case by case is beneficial. The evaluated framework consists on a two-branched neural network that contains both elements for row-wise correlation and pattern identification, as well as a temporal element. This structure was adapted from our previous work on continuous inclination prediction [10, 12], made target attribute agnostic, and the whole framework is open sourced and published on Github [13]. The case study of predicting inclination values between the sensors and the bit is used throughout this paper to evaluate various pre-processing configurations.

Additionally, data selection techniques were developed that are used to automatically find useful data range in the provided dataset. This involves automatic differentiation between small gaps that exist due to varying sampling frequencies and can be easily filled in, and big gaps, that exist due to equipment failure or change. The main contributions for that portion are:

4. Algorithm for automated identification of data gaps of varying significance is proposed.(Data Issue Identification)

5. Framework for RNN implementation in drilling is proposed to formalize the process, with focus on best practices for data pre-processing. (ML Process Framework's Formalization)

B.1.3 Paper Structure

After introduction (*section 1*), this paper presents a process framework (*section 2*) where individual pre-processing steps are described, as well as the model structure and the way the model is applied and evaluated. After the process is established research towards individual pre-processing steps is presented, focusing on data imputation (*section 3*), data resampling (*section 4*), and attribute selection (*section 5*).

B.2 Process Framework

This paper proposes a process framework designed for data prediction for sequential data logs focused on semi-automatic data processing. This allows to test approaches to data processing in a uniform rule-based manner easily, while keeping everything else equal. For example by automating the approach for attribute selection at different stages of well drilling, different attributes will be selected and therefore model dynamically adjusted.

The presented process attempts to automatically predict selected attribute from a selected raw dataset. Individual steps are listed in Table B.1 as well as presented in a flowchart form in Figure B.2. The process is generally split into four major elements: data selection, data pre-processing, machine learning model development, and its evaluation and inspection. First, data selection, focuses on automated identification of a usable area of the log in terms of completeness. This step is necessary as the framework aims at processing existing drilling logs;

real-time data often has gaps and unusable regions that have to be avoided. Pre-processing section handles problems like imputation, resampling, train/test split and attribute selection. Exploring configurations of this section is the key element of this paper. Machine Learning application section, while can consist of any regression algorithm, in this paper and study was equipped with a model that takes into account not only attributes row-wise, but has a temporal element, a recurrent neural network to capture the dynamic behaviour of the predicted attribute. Lastly, the evaluation and inspection section focuses on result analysis both in terms of presenting results from continuous application of the algorithm through emulated log creation while drilling (continuous learning), as well as data-driven sensitivity analysis [14]. Process presented in this paper focuses in principle on the research-workflow, and not field implementation; it is however designed such to simulate the field deployment and it is trivial to adjust it for such purpose. For a complete continuous learning approach it is necessary to repeat the complete workflow with different continuous Train/Test splits performed in the step 2.1. Individual steps are elaborated on below with subsection number matching the steps listed in the table, while research on specific areas of the framework is presented separately in the following sections.

B.2.1 Data Selection

B.2.1.1 Import raw dataset

Typical raw drilling data log consists of columns, designating attributes logged, and rows as the consecutive measurements. Logs often contain multiple missing readings. Most common format for data series is a comma separated values (CSV) file. For the framework to work a prediction target attribute has to be selected as well as the index, which in drilling logs is typically time or measured depth.

This paper uses the Volve dataset [2] for all presented calculations. It was published in 2016 by Equinor and it covers seismic, drilling, production and additional data related to the Volve field that is located in the North Sea. The

Table B.1: Process framework steps. Areas with key contribution of this paper are marked in **bold**.

1	**Data selection**
1.1	Import raw dataset
1.2	Drop unwanted columns
1.3	**Run gap statistics**
1.4	Identify and select longest stride
1.5	Remove incomplete columns

2	**Pre-processing**
2.1	Split dataset into continuous Train/Test dataset
2.2	Imputation
2.3	**Resample the dataset**
2.4	Scale dataset to (0,1)
2.5	**Select attributes/Principal Component Analysis (PCA)**
2.6	Shape the dataset to fit the model

3	**Applying ML model**
3.1	Take out part of Training dataset for Validation
3.2	Train the model
3.3	Stop when validation loss stops dropping
3.4	Calculate error on Test dataset

4	**Evaluation and Inspection**
4.1	Results analysis
4.2	Sensitivity analysis

dataset is fully open for everyone to explore and therefore it was used as a basis for case studies in this paper. Whenever well name is mentioned in this paper it is possible to acquire raw logs, daily reports, well plans, production reports etc. for this well.

Data used in this study was purely numerical, ie. there were no status attributes logged as text. If one is to apply presented methodology to a dataset containing such data, standard machine learning methods of converting such attributes to numerical values, such as one-hot encoding, should be applied.

In this paper the term *raw dataset* is used to indicate data not processed in any particular way, however if data is for example noisy an initial filtering and outlier

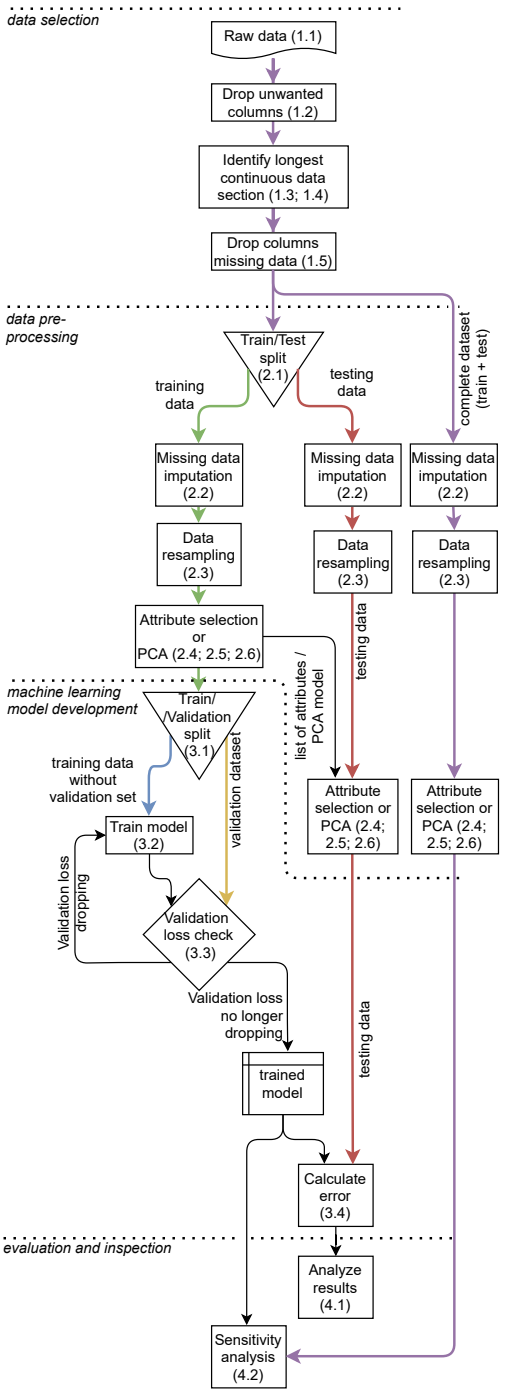


Figure B.2: Process framework flowchart, color coded datasets

removal is recommended. This is a very wide topic, and as this paper does not propose any new methods nor identify particularly useful ones, it is prudent to evaluate the data at hand and apply appropriate methods. Basic filtering such as rolling mean is a typical good starting point and methods such as data validation and reconciliation [15, 16, 17] are worth considering.

B.2.1.2 Drop unwanted columns

It is important to remove attributes derived from the target through simple math. Most basic and common example would be predicting *ROP* while *Inverse ROP* also exists in the dataset, which is calculated based on the ROP. Leaving the *Inverse ROP* as an input for predicting *ROP* would defeat the purpose of the model and therefore *Inverse ROP* parameter has to be manually identified and removed. This is the case in the Volve dataset, where attribute *Inverse ROP (5ft avg)* is often present.

It is important to highlight that the attribute availability is a part of the architecture of the designed process and should not be confused with attribute selection for the ML model. Attribute availability is dictated by *what* is to be achieved, and attribute selection is related to *how* it is being achieved. For example, if a problem at hand is to predict inclination data that is delayed due to sensor position it is unreasonable to provide the model with vibration or azimuth data recorded by the same device, as it is also delayed and hence unavailable. Only surface recorded parameters, such as ROP, rotary speed (RPM), weight on bit (WOB) etc. will be immediately available. However, if the task at hand is to recover inclination data due to this specific sensor failing at some point, it is valid to include vibration and azimuth data.

In the presented case study of inclination prediction the following columns were dropped:

- *RHX_RT unitless* - magnetic sensor raw data
- *RGX_RT unitless* - gravity sensor raw data

- *MWD Continuous Azimuth dega* - azimuth

The azimuth data was removed as it is delayed due to the same reasons as inclination data that is to be predicted. Raw sensor data also exists in the dataset for both inclination and azimuth, which has to be removed, as the target is a simple function of those, also delayed, readings. Additionally index counter was removed, as well as columns containing single constant value, such as well name.

It must be noted that some columns that are fully static within a well, f.ex. *well name*, can be useful for the model if multiple wells are fed into the model. In case of categorical data a one-hot encoding should be utilized, that is an attribute *well 1* and *well 2* would be introduced that would have a binary value of either *1* or a *0*. Such static attributes can be processed by the proposed workflow the same way as other data.

B.2.1.3 Run gap statistics

Data can be missing from the raw dataset for various reasons. It can by typically understood that data gaps are either small, created due to uneven or out of sync logging, or big and generally not easily recoverable that exist due to equipment change or failure. It is therefore necessary to evaluate if a given gap in attribute readings is small enough to be imputed using simple methods such as forward filling (last valid value forward) or linear interpolation, or the gap is too large such that it has to be abandoned. Our own novel method of analysing data gaps, named Gap Coefficient (GC) is proposed here, where the gaps are classified based on their continuous length and the percentage of the dataset that they occupy. It is proportional to the gaps' continuous length and inversely proportional to the quantity of gaps of a given size and the total length of the dataset. This way small, but plentiful gaps return low value, while few big gaps return high value. The GC is calculated as:

$$GC_i = \frac{i}{GQ_i \cdot TL} \tag{B.1}$$

Where i is the gap length, GC_i is gap coefficient calculated for the gap length of i, GQ_i is the quantity of gaps of length i and TL is total length, or row quantity of the whole dataset. Visualization is provided in Figure B.3. For a more realistic example, when evaluating gap length of 5 rows, if there are 50 gaps in a dataset of total 4000 rows, calculation will be:

$$GC_5 = \frac{5}{50 \cdot 4000} = 0.000025 \tag{B.2}$$

All gap sizes for all attributes are calculated separately. A cutoff value δ_{cf} can be selected based on experiences, cases and requirements, where $GC > \delta_{cf}$ signifies a gap that should not be infilled by forward filling or linear interpolation as it is too big and not caused by typical small-gap processes as asynchronous logging or varying logging frequencies. In general, small and plentyfull gaps will yield a low GC value, while few big gaps will return a high GC. In our case studies, δ_{cf} is set as 0.005, but for other datasets different value may be more appropriate..

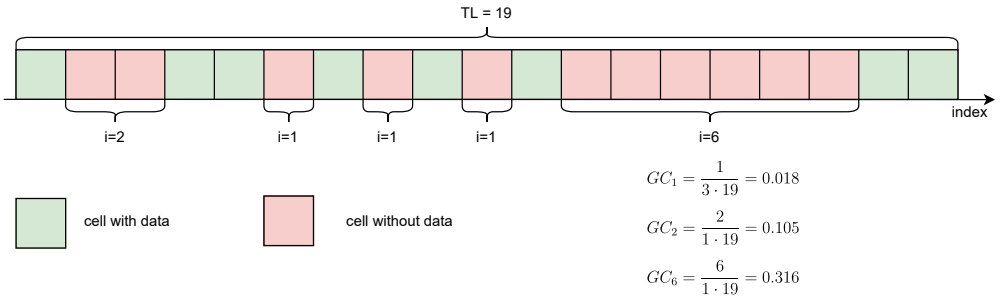


Figure B.3: Gap coefficient calculation example

B.2.1.4 Identify and select longest stride

By using the GC, it is easy to calculate usable or unusable areas of the log for each attribute. As the same process is applied to the target (predicted) attribute, it is possible to identify the longest part of the well log when this attribute has only small gaps, here called a *stride*. This is the part of the provided log that will be used for training and evaluating the model.

B.2.1.5 Remove incomplete columns

As the longest stride for the target attribute is identified, it is possible to select compatible attributes that have complete data in the same area of the log. Small, one percent margins are cut off from the target attribute stride at the start and at the end. This is done because target attribute might appear a small distance before others, which would lead to unnecessary attribute removal. This value may be tweaked to retain maximum amount of data.

Additional check is performed on the first derivative of the index attribute (typically depth or time) to ensure continuity of the log. If there are any outliers, for example a handful of very high values in the first derivative of the index, it suggests that the log is not continuous and should be manually inspected.

B.2.2 Pre-processing

In this section, we put our main focuses and discussions on data imputation and resampling, attribute selection by PCA and data scaling. Other important data pre-processing techniques for invalid data, noisy data, incorrect data, delayed data and data with outliers are out of this work focus.

B.2.2.1 Split dataset into continuous Train/Test dataset

To mimic real-life application the dataset is split into continuous sections for a given percentage of well drilled. This is what this paper refers to as *training while drilling*, where training dataset is continuous section from *0%* to *x%* of well drilled, while testing dataset is from *x%* to *(x+F)%* of the well drilled, where F designates the quantity of the well to be evaluated at a given step; in the case studies used in this paper $F = 20\%$. This paper typically used $x \in (15, 80)$ in 50 evenly spaced steps for a full evaluation. The lower boundary of 15% was selected due to limited nature of the case study's dataset; predicting inclination relies on identifying the sliding and rotating sections, and these are needed in the

training dataset. First 15% contained just one such cycle. Note that when bigger datasets are available, or problem at hand is simpler, model can begin to show good performance earlier.

It is a common mistake in machine learning applications in data series, such as drilling logs, to split the dataset randomly into training and testing. It was shown that even random values, such as a random walk can be predicted with an R^2 metric as high as 0.9948 based on index alone if split randomly, as presented in earlier research [18], despite the model being clearly nonsensical. Random train/test split is common in other types of ML applications, but when dealing with data-series the sequential split, as presented in this paper, is the correct approach.

B.2.2.2 Imputation

Imputation for data series, in relationship to small data gaps, can be either done by forward filling (using last valid value forward), linear interpolation or other types of curve fitting methods. While many methods of imputation exist they are often either not suitable for drilling logs, such as filling with mean, or impractical for logs that contain a lot of missing values throughout the dataset, such as regression. There are also methods such as polynomial interpolation, that has potential for higher quality imputation, it will not work in some common cases, such as data gaps separated by a singular data point.

To decide whether forward filling or linear interpolation is better suited a check is performed on each attributes' first derivative, calculated while temporarily removing missing data. If more than 90%[1] of values of the first derivative are zero, then forward filling is selected as presumably the better method. This will identify rarely changing values such as wellbore diameter, which is more likely to change rapidly or discretely, where linear interpolation would introduce unrealistic slopes to the data. Selection of the threshold value is explored and discussed in detail in further sections of this paper.

[1] base case value, evaluated further in the manuscript

It is critical to perform data imputation utilizing linear interpolation after the division into train and test data. As the dataset is split into continuous training and testing portions infilling before a split would *leak* future information [19]. For example, if drilling torque suddenly rises and the data train-test split happens to be just before data row with the rise, linear interpolation would cause the value to rise due to effect in the future, making the model perform better than it realistically should in real-life operation. After the data split, linear interpolation can be performed without the risk of data leakage, or 'peeking', as the dataset at this stage is split into training and testing. Linear interpolation cannot fill missing data that resides at the edge of the dataset, and those values are filled using forward and backward filling.

B.2.2.3 Resample the dataset

Drilling logs are likely needed to be re-sampled before temporal machine learning models utilizing RNN can be applied. This is because the sampling rate is not constant leading to stretching and shrinking of the signal. It may also be desirable to reduce the quantity of data rows used for various reasons. Resampling will dictate the resolution of data, meaning what distance (or time) is between two consecutive samples. Resampling methods are discussed in detail and evaluated in further sections of this paper.

B.2.2.4 Scale dataset to (0,1)

Machine Learning algorithms typically work best when inputs are scaled to (0,1). There are competing approaches [20] to scaling, normalizing, or standarizing data, and these are outside of the scope of this paper. Simple (0,1) scaling is applied when *pearson* or *ppscore*, Predictive Power Score, is used further in the process for attribute selection purposes, as such scaling is transparent to these correlation methods. If Principal Component Analysis (PCA) [21] is applied as an alternative to attribute selection, the base case is also (0,1) scaling; however an option to apply standardization is provided; this process is transforming attributes such

that the mean becomes equal to zero and standard deviation becomes equal to one. This alternative is explored later together with attribute selection strategies. Note that the PCA implementation used here [22] automatically centers the data (but does not scale it) as this is a prerequisite for the PCA transformation. Therefore practical difference for two options above are either min-max based scaling or standard deviation based scaling. PCA is extremely sensitive to scaling and it is a key pre-processing element [23] that explored in the further section of this paper. Pre-PCA scaling cannot be omitted, because the resultant PCA component values would be overwhelmed by attributes with high numerical values.

B.2.2.5 Select attributes, PCA

Two methods of selecting attributes are implemented and evaluated: pearson coefficient [24], ppscore [25] which provides asymmetric correlation of coefficients based on decision trees. Additionally, Principal Component Analysis [21], a method for dimensionality reduction, can be applied that also achieves the goal of reducing the amount of inputs. These three approaches (pearson, ppscore, PCA) are evaluated in this paper. Despite the fact that (0,1) scaling was done already, if PCA transformation is applied the data is likely not to be within (0,1) range anymore, hence to improve machine learning performance (0,1) scaling is applied again.

B.2.2.6 Shape the dataset

The dataset has to be reshaped to fit the machine learning model, which means that samples are created that contain the input and predicted output of the model. A user shall specify the length of data of the target attribute, say 25 meters in the presented case study for inclination prediction, is used as input to the recurrent neural network processing the temporal information of the signal. This paper refers to this as *memory area*. Further, directly adjacent length of target attribute, in the case study 25 meters as well, is selected for prediction, which this paper refers to *prediction area*. Other attributes from the dataset

residing in the prediction area are also selected as input of a single sample. For better visualization of the sample shape Figure B.4 is presented, where memory length is set at 5 samples, prediction length is at 4 samples. Predicted attribute is A with B, C and D as additional attributes.

In conclusion, a single input sample consists of $\mathbf{n} + \mathbf{m}$ values of target attribute, where n is the row quantity from the memory portion and m is the row quantity from the prediction area, as well as $m \cdot j$ values, as there are $\mathbf{m}$ rows in the prediction area and $\mathbf{j}$ attributes, which are all selected attributes from the dataset (or PCA dimensions). The exact quantity of $\mathbf{m}$ and $\mathbf{n}$ rows will depend on the length of the prediction and the memory area (defined by user), as well as re-sampling rate. In Figure B.4, $m = 4$, $n = 5$ and $j = 3$.

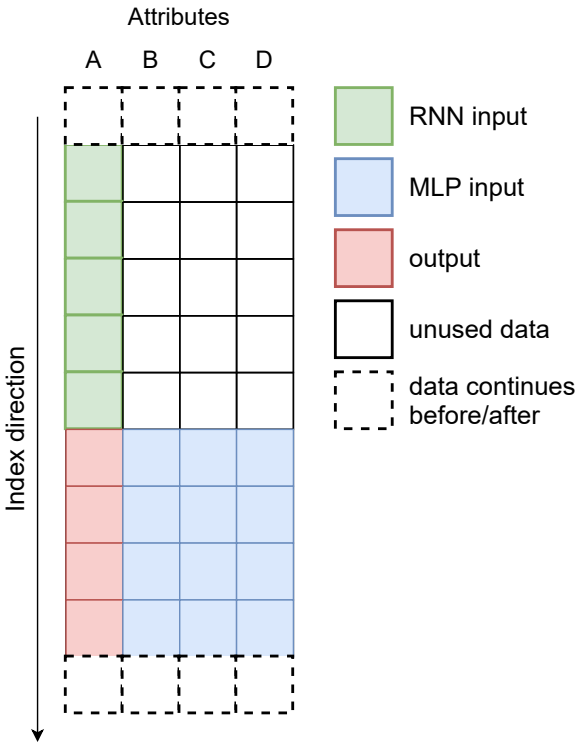


Figure B.4: Shape of the data showing RNN and Multilayer Perceptron (MLP) input shapes

B.2.3 Applying ML model

B.2.3.1 Dataset for validation

In the use case of *training-while-drilling*, or *continuous learning*, where the training dataset is created as the well is being drilled, the amount of data is quite low, especially in the initial model training phase using continuous training concept. This often leads to overfitting of the machine learning model [26]. To tackle that a validation dataset is taken from the training portion of the data. This may also be referred to as *masking*, where a portion of data is masked, and therefore unavailable for training even though it is technically available to the user. By monitoring the error on the validation dataset it is possible to pinpoint the moment when the model is no longer learning, but begins to overfit. This technique is called early stopping and is a recommended practice, especially for small dataset [26]. In our case study, the validation portion consists of 15% of the available training data taken from the newest available samples. These values will change depending on the problem at hand and computational power available in deployment.

The approach of masking portion of available dataset for gauging the models' performance is commonly used in ML and applied in petroleum related topics [27, 28].

B.2.3.2 Model training

Model takes into account both temporal information about the target signal as well as attribute values from the same index rows as the prediction. For example, when predicting the ROP with taking into account the WOB and RPM the model will predict the ROP in the prediction area based on how the ROP changed in the past, as well as the current WOB and RPM. Simplified model structure is shown in Figure B.5. Memory (n) and prediction (m) step quantities are selected by users; the case studies presented in this paper typically use $n = 100$ and $m = 100$. This, connected with selected resampling rate for the case study of

0.25m between samples, results in the memory and prediction being 25 meters long. Right hand side model branch consists dense layers making a Multilayer Perceptron (MLP) [29] artificial network, while left hand side branch consists of Recurrent Neural Network employing Gated Recurrent Units [6] (GRU). While in the presented case study and evaluation further in the manuscript GRU was selected as as the RNN due to its performance on relatively small datasets, it is trivial to substitute it with Long-Short Term Memory (LSTM). Pure RNN layers are not recommended as they suffer from the vanishing gradient problem [30]. Technical details, such as specific Keras implementation, numbers of neurons in all the layers, activation function, etc. are provided on Github[2].

To visualize and better convey the configuration of inputs and outputs Figure B.6 is shown. It contains complete data related to one randomly selected sample. Data sharing same x-axis location is related to the same physical location in the well in relation to the bit. Inputs are shown in black; in this example there is one RNN input (here M*WD Continuous Inclination*, in degrees), and 5 PCA components as MLP inputs. Dotted blue line shows true inclination for the sample and dashed red line shows the prediction generated by the ML model. Note that in practice all approximately 50 attributes were used as inputs to the PCA here, and it is possible to consider the ML model to be inclusive of the 50 physical attributes. This however would be difficult to visualize and not representative of data fed to the neural network itself.

B.2.3.3 Stop when validation loss stops dropping

While training the model validation loss is continuously measured. If the validation loss at a given epoch is lowest since the training began, the model is saved for use, but the training continues. Through consecutive iterations the validation loss normally drops continuously, until it levels off and begins to rise, signifying overfitting; this procedure is called early stopping [31] and is widely used in machine learning. A specified number of epochs is set as *patience*, which defines

[2]https://github.com/AndrzejTunkiel

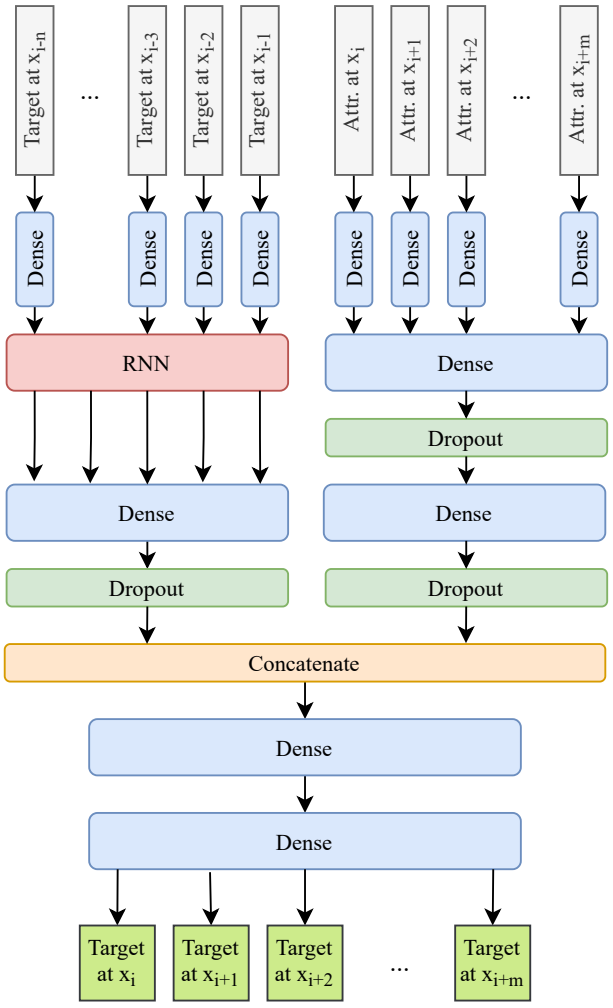


Figure B.5: Simplified model schematic

the amount of epochs since the last validation loss improvement is allowed. In presented implementation patience of 50 epochs was used.

B.2.3.4 Calculate error on Test dataset

The score of the model is calculated on the test dataset, that remained untouched throughout the training process. To completely evaluate a given model it is critical to repeat the training process, each time starting with an untrained

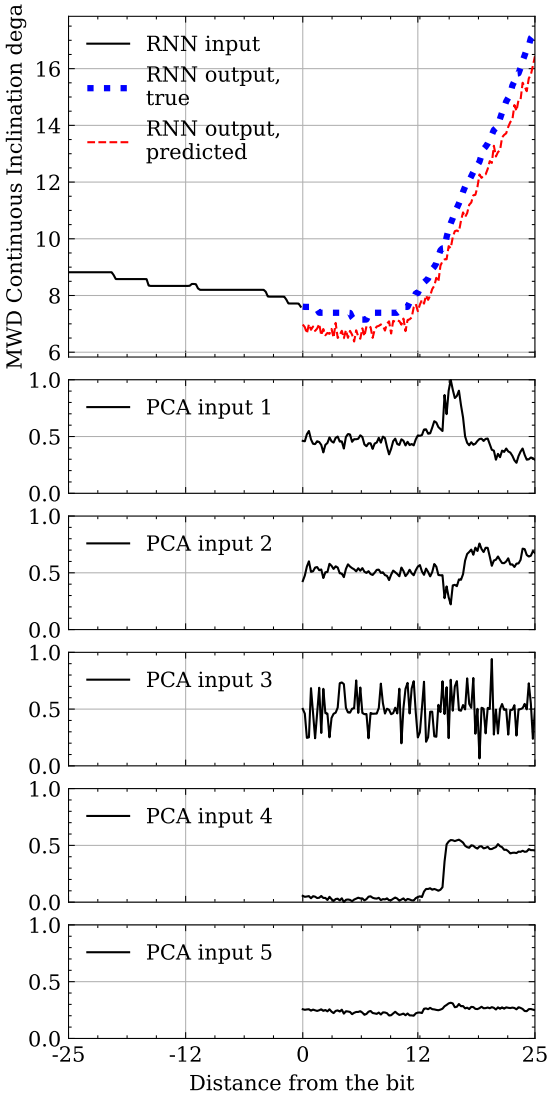


Figure B.6: Example of inputs, outputs, and predicted values. Inclination prediction case study.

model, for multiple points in the dataset simulating continuous drilling of a well, ie. *training while drilling*. Proposed framework uses mean absolute error (MAE) as a key metric that is used throughout the paper. Other metrics, such as R^2, weighted mean absolute error, average percentage error may be better suitable for specific case studies, however scoring method selection is outside of the scope

of this paper.

B.2.4 Model Evaluation and Inspection

B.2.4.1 Results analysis

Results are evaluated by incorporating training-while-drilling methodology [10] derived from a continuous learning concept [32]. In this methodology it is assumed that the machine learning prediction model will be trained while drilling, hence the performance of such model has to be explored with consideration of continuously expanding training dataset available. Figure B.7 shows how the training and testing datasets are split in relation to the bit position and which data is considered available. As the drilling progresses, more data is available. A fixed portion of the data, here 20% is used, that is at the end of the drilled section is used for testing.

This case study and train-test split is used throughout the paper. A sample result is shown in Figure B.8, where the error is shown as a heatmap; the color designates mean absolute error and is a function of the percentage of well drilled and the prediction horizon. Each row in the heatmap designates performance of a model trained after reaching a given depth, shown on the y-axis together with the percentage covered of the dataset at hand. The x-axis designates the location of the prediction, with 0m being at the sensor, and the bit being 23 meters away from the sensor. 20% of the dataset directly adjacent to the training data is used for testing and calculating the error. When this paper discusses results as a single value, an average of all prediction points from all steps in training-while-drilling process is used.

It is important to understand what the presented model predicts to understand the results fully. It was developed in tackling the sensor lag problem [12], where, referring to Figure B.9, the information is not yet recorded for a portion of the well due to sensor position. Inclination data is unavailable ahead of the sensor, but is available behind the sensor, and at the same time real-time attributes,

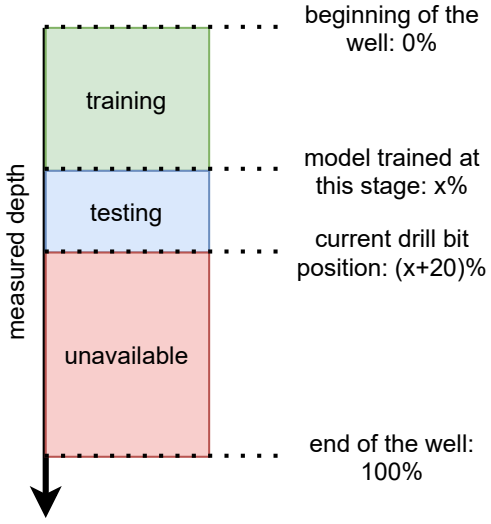


Figure B.7: Training-while-drilling, train-test data split example.

such as ROP, Weight on Bit, Surface Torque, etc. are available for the complete section. The results are generated for the area on the distance marked **b** using **m** continuous datapoints, or steps; data in the distance **p** containing **n** samples is considered available.

When applying the model in practice there are multiple metrics that may be interesting depending on the actual problem at hand. For example, for the inclination prediction model, while based on the inclination value in degrees, one of the developed metrics was the predicted position of the bit in (x,y) coordinates later evaluated as using R^2 value [10].

B.2.4.2 Sensitivity Analysis

Sensitivity analysis of the model can be performed using Data-driven Sensitivity Analysis methodology [14], where the model is made based on the complete dataset and then the complete dataset is used as a starting point for one-at-a-time sensitivity study. The results, due to multiple starting points, are considered

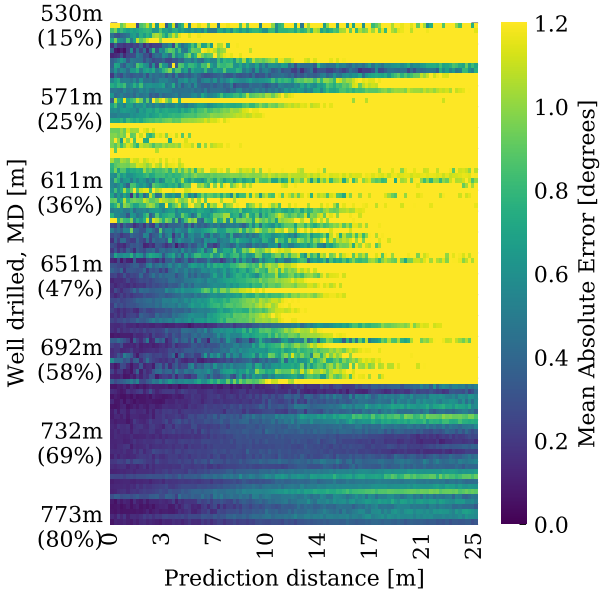


Figure B.8: Example results of continuous learning applied to predicting 25 meters ahead of the bit.

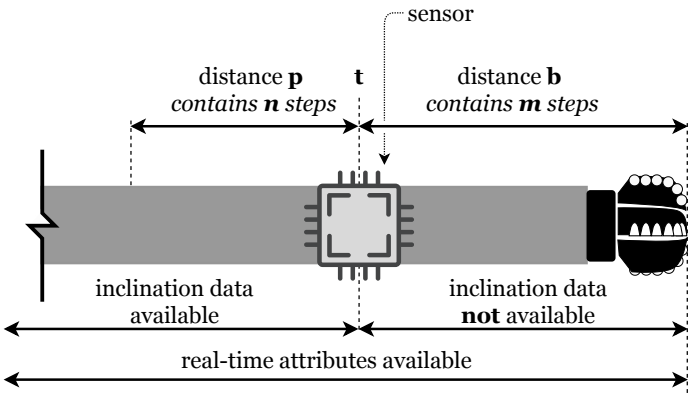


Figure B.9: Sensor lag case study.

statistically. More traditional methods, such as sobol indicies [33] can also be applied.

As this topic is very broad it is not evaluated in details in this paper, only presented as one of the elements in the complete workflow. Multiple sensitivity analysis methods exist and it is beneficial to perform such analysis when a model

is explored, to gain more insights into the working of the model, as well as to identify potential issues that would otherwise remain unidentified.

B.2.5 Hyperparameter Tuning

In general, different problems will require different hyperparameters for a machine learning model to perform at the peak performance. The process of identifying hyperparameters such as neuron count for individual layers, activation functions, dropout rates etc., called hyperparameter tuning, is necessary before finalizing the model. In this paper, the hyperparameters were tuned for a case study of predicting continuous inclination with 25 meters of memory and 25 meters of prediction, PCA dimensionality reduction and moving inclination data in individual samples to local coordinate system. When evaluating different aspects of the model, such as re-sampling algorithms with all their parameters, it is not feasible to perform hyperparameter tuning for all possible combinations to establish true error minimums. Introducing changes however will indicate the system's sensitivity to those changes, and will be indicative if certain configuration is better than the other without the need of performing full hyperparameter tuning.

B.3 Data Pre-processing: Imputation

As indicated earlier, the imputation in the data series when related to real-time drilling logs is often omitted despite being a common practical problem. This paper tackles two separate issues: big gaps covering significant stretch of the measured process, and small gaps resulting from varying sampling rates of measuring equipment. The other distinction between these two gap sizes is the fact that small gaps are assumed to be imputable with simple techniques of forward-filling, otherwise known as *last observation carried forward*, and linear interpolation, while the data in the big gaps is considered unrecoverable, at least using simple methods.

Identification of big gaps in the process this paper uses is done as the one of the first steps after importing data, as this will determine the usable range for further model evaluation. This is done automatically based on the selected target attribute using proposed Gap Coefficient from Equation (B.1)

$$GC_i = \frac{i}{GQ_i \cdot TL}$$

This simple equation is used to calculate an indicator value, that is proportional to the relative size of the gap and inversely proportional to the quantity of gaps of the given size. Basic statistics related to gap sizes and the percentage of dataset that they occupy are shown in Figure B.10 for MWD Continuous Inclination attribute of Volve well F9a, depth based data. In this specific case, under 12% of cells in the raw log contains data, and the rest is empty. Gaps are of various lengths from 1 row to 1480 rows. Note that the row here refers to a raw dataset that is sampled unevenly; in the case of well F9a the mean distance between the rows is 0.05m.

Proposed GC data is presented in Figure B.11. The cutoff value $\delta_{cf} = 0.005$ is proposed in dash-line; this value should be considered a tentative proposal for real-time drilling logs, as the optimal value might be different for other processes. Once the GC is above such cutoff value, the data section for a given attribute is considered unusable due to the large gap.

When *big gaps* are identified it is possible to identify continuous log portions consisting small gaps only that this paper calls *strides*, see section B.2.1. In the proposed automatic process the longest stride is used for analysis by default. This approach has a risk, where an attribute significantly correlated with the target, and therefore highly useful for prediction, exists only in a portion of the stride, which would exclude it from analysis. This is a limitation of the proposed semi-automatic approach and such cases have to be identified and mitigated manually.

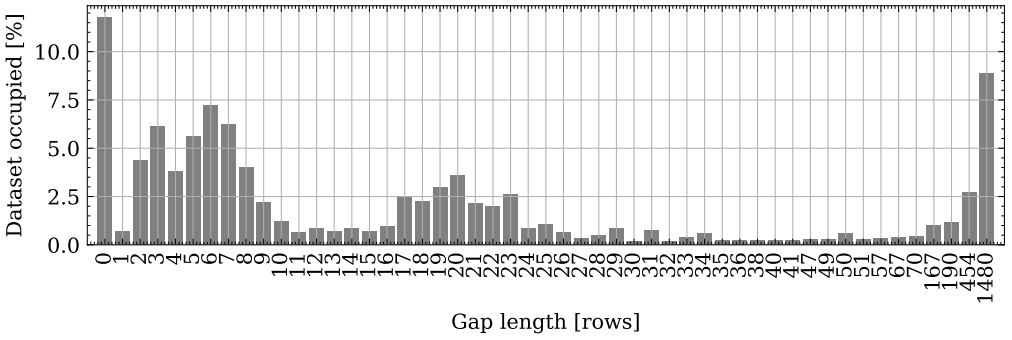


Figure B.10: Raw gap statistics, MWD Continuous Inclination, Volve well F9a.

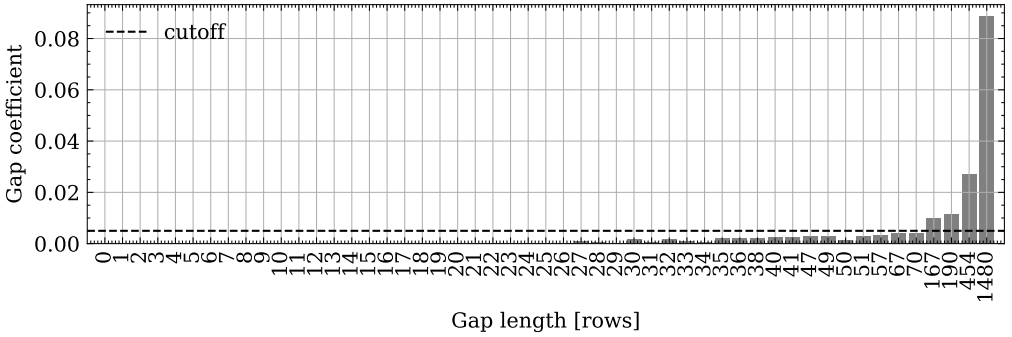


Figure B.11: Gap statistics, gap coefficient, MWD Continuous Inclination, Volve well F9a.

An algorithm is proposed to determine if a given attribute should be infilled (in relation to small gaps) via forward filling or linear interpolation. The working assumption is:

Assumption 1: *if an attribute is mostly stationary the small gaps are best filled with forward filling. Other parameters will be filled by linear interpolation.*

Attributes such as nominal wellbore diameter, on-bottom flag, or stick-slip flag, would be examples where forward filling is best, as it does not change often throughout the well, and is unlikely or impossible to change gradually. The validity of this assumption is presented in further sections, where *imputation algorithm selection threshold* is evaluated

Rationale behind considering only forward filling or linear interpolation is such, that those algorithms are extremely robust in terms of the imputation. They

only rely on singular data point at one or both ends of the gap and therefore easily fill all the gaps, with an exception of linear interpolation failing to fill gaps at the start and the end of the data series; this is fixed by applying forward fill and backward fill after linear interpolation to close those gaps. Furthermore, a correctly setup measurement system works under assumption:

Assumption 2: *the last received measurement is valid, or the polling rate is fast enough to capture the nature of the signal by being above Nyquist rate* [34].

If the assumption on the Nyquist frequency is invalid, then signal will exhibit artifacts, similar to Moiré patterns, where poorly resolved high frequency signal generates a low frequency *artifact*. In such case the presented gap infilling will further copy the apparent signal. This is however in principle logging system failure, not a limitation of the presented methodology. Statement on the validity of last received measurement is generic, meaning that values in dataset are considered correct at stated index position.

To identify if an attribute falls under forward filling or linear interpolation algorithm, this paper proposes to use a threshold ($\theta_{th}\%$) as an selection criterion by calculating the relative quantity of zero-valued numerical first derivatives of a given data series, and therefore identify generally stationary and generally non-stationary signal. For example, a data series with attribute X with $k+1$ data points is given as

$$X = \{x_0, x_1, \ldots, x_k\}.$$

Let us define a new calculated dataset

$$\Delta X = \{\dot{x}_0, \dot{x}_1, \ldots, \dot{x}_{k-1}\},$$

and n_o is the number of zero-valued of points in ΔX. The algorithm is that

- Forward filling is applied if $\frac{n_o}{k} \geq \theta_{th}\%$;
- Linear interpolation is used when $\frac{n_o}{k} < \theta_{th}\%$.

In the case study, to identify optimal value a grid search simulation was applied, where a full training-while-drilling scenario for inclination prediction consisting of 50 train-test iterations was applied to predict continuous inclination of well F9a of

the Volve dataset. Prediction distance was set for 25 meters and 100 samples, and the memory was also 25 meters and 100 samples. The threshold was varied from 0 percent to 100 percent by 10 percent increments. After running the simulation 23 times for each of the thresholds totalling 12 650 train-test iterations, the mean absolute error was analysed, with results presented in Figure B.12. Multiple simulations were performed for the same configurations since the training process is stochastic in nature, and re-running it with different random seed will yield different results.

Results suggest that in this case study threshold should be at least 20%, where the MAE is lowest. Since the spread in MAE values is high, a t-test was applied to verify that results are significant. The difference between the worst score at 0% (always forward filling) and best at 20% is significant with $p - value = 2.6 \cdot 10^{-7}$, and the difference in the MAE is 6.5%. Difference between the threshold of 20% and 100% is also significant at $p - value = 2.2 \cdot 10^{-5}$. This shows, that for at least some cases, there is a significant difference between infilling methods applied for small gaps and that using only one or the other method is not optimal with selection based on first numerical derivative a good alternative.

It is also worth indicating, as shown by the attribute count subplot in Figure B.12, that drilling attributes in a drilling log generally fall into one or the other category of filling no matter where the threshold is set between 10% and 90%.

Another study was run, this time for ROP prediction on the same dataset, well F9a, and otherwise generally the same settings, corrected for attribute removal (f.ex. removed *Inverted ROP* attribute for this case study). In this case, shown in Figure B.13, the imputation based purely on linear interpolation for all the attributes showed the best results. T-test between pure forward filling and linear interpolation shows that results are significant at $p - value = 5.3 \cdot 10^{-5}$ for a mean 1.6% difference between them. These results show that selecting the optimal filling method may bring a modest, yet statistically significant performance improvement, and it is worth considering when performing hyperparameter tuning.

While performance improvement was expected, it is difficult to find the exact reason for performance change between specific settings. It is also interesting that while for inclination prediction case study linear interpolation for all attributes performed poorly, for ROP prediction it was a reasonable strategy. More research is needed to better understand results shown here. Simpler machine learning models may be better suited for such task.

Remark 1: Note that hand-picking and/or evaluating imputation method for each and every attribute is likely to yield superior results. Presented method acts as an automation method that can yield modest prediction quality improvements.

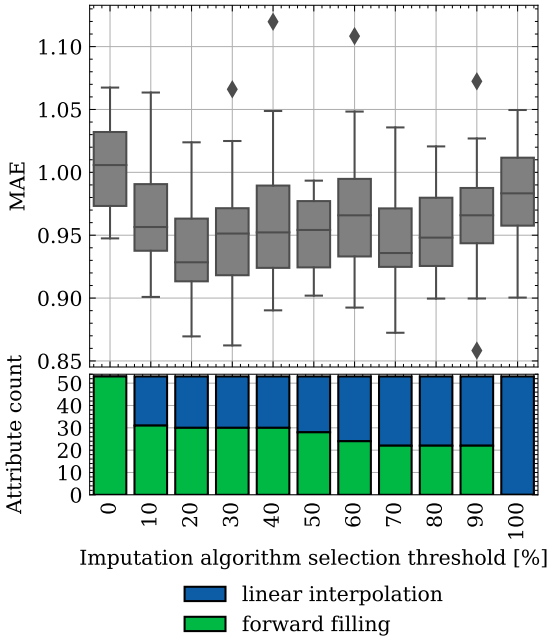


Figure B.12: Smart filling study, inclination prediction, by varying the threshold for applying linear interpolation versus forward filling, the percentage of zero-valued first numerical derivative values. (0% - Forward Fill only, 100% - linear interpolation only)

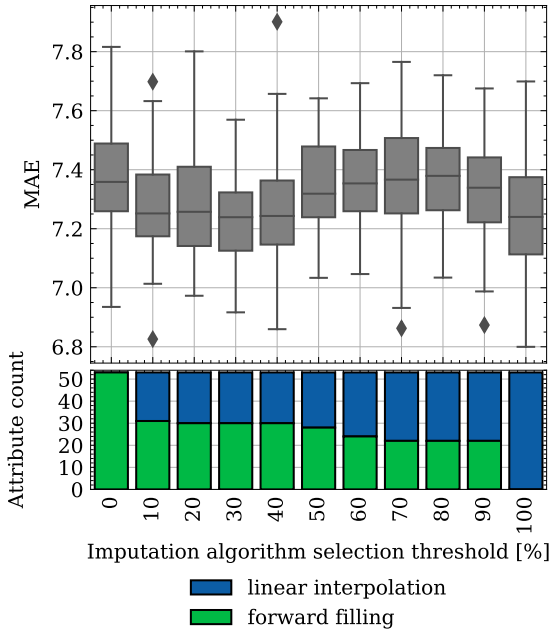


Figure B.13: Smart filling study, ROP prediction, by varying the threshold for applying linear interpolation versus forward filling, the percentage of zero-valued first numerical derivative values. (0% - Forward Fill only, 100% - linear interpolation only)

B.4 Data Pre-processing: Resampling

When utilizing the RNN, similarly to other signal processing methods, it is implied that the steps between the consecutive data points are evenly spaced. While there is recent research related to modification of RNN architectures to introduce time gate [35] that mitigates the issue, the asynchronous nature of recorded value remains a practical problem. Additionally, the RNN architectures implementing such time gate are not included, as for now, in the most common machine learning libraries such as Tensorflow/Keras or PyTorch.

In typical real-time drilling logs the sampling rates vary and the readings are asynchronous. Even if readings are evenly spaced in time domain, they will not be such in the depth domain and vice versa. Common technologies, such as mud-pulse telemetry, transfer readings from multiple sensors sequentially, and

therefore out of sync due to low bit-rate of this data transfer technology. Therefore re-sampling of the data logs is critical pre-processing step when working with drilling logs.

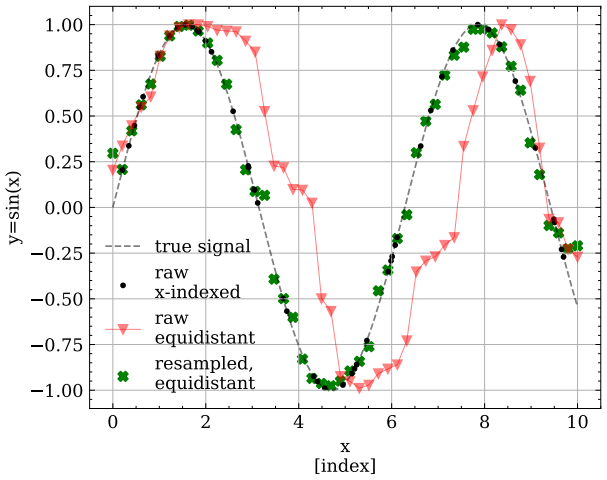


Figure B.14: Importance of resampling, example of signal deformation due to uneven sampling rate

To re-sample a dataset it is to change it such that the consecutive sensor readings are recalculated for arbitrary index values. For example, when working with a depth based dataset, original ROP readings may be (50, 59, 61, 70) *m/h* logged for Measured Depth (MD) of (100, 104.5, 105.5, 110) *meters* respectively. When looking at the raw ROP values it is obscured that the ROP grows close to linearly in relation to the MD. In this simple example the log can be re-sampled for MD of (100, 105, 110) *meters* resulting with ROP at (50, 60, 70) *m/h* respectively. While in this case re-sampling reduced the number of data points from 4 to 3, the information is much clearer. If the same log has the WOB readings (30, 40, 35, 35, 35, 40, 40, 35) *klbs* at corresponding MD of (98, 100, 102, 104, 106, 108, 110, 112) *meters* re-sampling is again necessary to bring the reading in-sync with the other attributes, even though the sampling rate is even.

B.4.1 Resampling Importance and Algorithms

In practice, to perform re-sampling it is necessary to create a regression algorithm to estimate the output variable value (an attribute) for an arbitrarily selected input (index) variable, which in the case of drilling logs is typically either depth or time.

To further visualize the problem Figure B.14 is produced to better explain the issues at hand. First a sine function is applied to values of x between 0 and 10. True sine function is plotted as a dashed black line. 50 random points in (0,10) were selected from a uniform distribution and the true value of sine was calculated for those point; these values are plotted as black dots and represent *non-uniformly* sampled sensor readings. If the actual index value for those readings is ignored and they are plotted as 50 *uniformly* distributed points the nature of the sine function is greatly distorted, see red triangles plotted in the same figure. In numerical signal processing it is common that the index of a signal is not retained explicitly for each data point and assumed constant throughout the dataset. This, in presented basic example, heavily distorts the temporal nature of the sine function and is detrimental to achieve good results with recurrent neural networks. It is however possible to re-sample the data to match the original function much more closely. In Figure B.14 such data is calculated using K-Nearest Neighbor (KNN) regression [36] algorithm for neighbour count of 3 and distance weighted; results are plotted as green crosses. The results are not perfect, but are a significant improvement when compared to no re-sampling, plotted in red.

This paper explores two algorithms that are particularly suited for re-sampling: KNN [36], and Fixed Radius Neighbour (FRN) [37]. The two algorithms are similar to each other. Given a two dimensional dataset, an arbitrary value along the x-axis can be selected and the algorithm will identify K closest points (KNN) or points within the specified radius r (FRN) and return their average value. Additionally, it is possible to assign weights to the identified points based on the distance from the point of interest along the x-axis, where individual weights

w are equal to inverse distance d, $w = 1/d$. For the purposes of this paper scikit-learng implementation of these algorithms was used [22].

B.4.2 Resampling Quality Evaluation

B.4.2.1 Known ground truth

It is, in principle, not possible to directly evaluate the performance of resampling algorithms on drilling data, as the ground truth is not known. It is not possible to calculate the difference between resampled data and true data, because the samples do not share the same index values. True values simply do not exist in the new index positions in the raw data. Therefore as a first step evaluation using a sine wave was performed. Values of sine wave can be calculated for any value to check the resampling quality against the ground truth. The dataset was created the same way as shown in Figure B.14, where 100 points along the x-axis were generated instead of 50 to not exaggerate the errors. Two algorithms were evaluated, the KNN using K-Nearest Neighbours Regressor implementation and the FRN using Radius Neighbours Regressor, from scikit-learn implementation [22]; two weight options were tested, *uniform* and *distance-based*, giving four different combinations. For the KNN algorithm the k-value was evaluated from 1 to 50, and FRN radius multiplier from 1 to 10, with the base radius being maximum distance between neighbouring data along x-axis.

Since the test was based on randomly generated values along the x-axis, a Monte Carlo simulation [38] was set up with 100 runs for each combination. The results are presented as the mean absolute error between the re-sampled data and true value of sine function. Additionally, as a reference, the error value for non-resampled data is provided from a case where non-uniformly distributed sine function data is evaluated against uniformly distributed values of the sine function. Results are presented in Figure B.15.

Inspecting the results it is clear that the lower the radius and the k-value, the better the results. There is slight exception in case of KNeighborsRegressor used

with distance weights, where the lowest neighbor count registers an uptick in error. In general the results are in line with intuitive expectations, where the best correlation with reality is for methods inspecting the closest vicinity of the re-sampled points, that is close to the smallest radius or lowest quantity of neighbors.

While synthetic results can be indicative of real-world performance, it does not mimic the reality fully. While in principle all signals are a collection of sine waves that can be decomposed thorough a Fourier transformation, it does not necessarily indicate that results from a single sine wave will match any arbitrary signal.

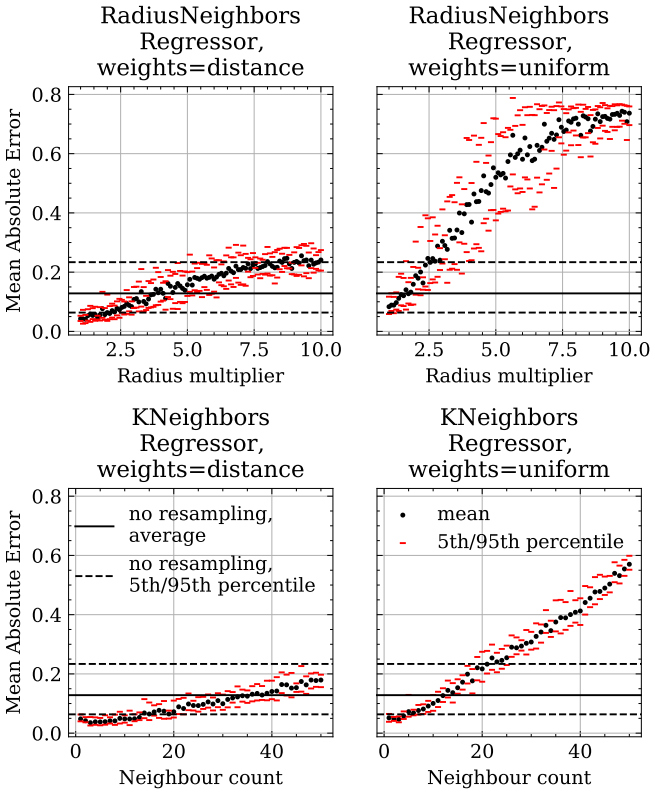


Figure B.15: Resampling error study

B.4.2.2 Unknown ground truth

To calculate how well the resampled signal matches the original data, where the ground truth is not known, a method based on integrating squared difference between data is proposed. Let's assume that $g(\cdot)$ is the difference between the function describing original raw data $r(\cdot)$ and a function describing resampled data $t(\cdot)$, or

$$g(x_i) = r(x_i) - t(x_i), \tag{B.3}$$

where x_i is the data variable (point). In practice those functions are defined by tabulated data, hence let those functions be defined as straight lines between two neighboring points, see Figure B.16.

It is trivial to calculate the integral of $g(x_i)$ as an area of a set of polygons, however this is not a robust way of evaluating the goodness of fit, as it in practice uses linear weight for the error, while a square error is typically preferred, to penalize few large differences more than many small ones. This is similar to a technique where the mean square error is evaluated against a rolling average in function of the window length, however the task is more complicated in the case of data resampling since the index of original and new datapoints don't match. To solve this issue the squared difference between the two functions is calculated discretely.

The practical implementation is based on Riemann sum [39]. A basic example of the method for calculating the discrete difference between the curves is shown in Figure B.16. Note that the index of the data points do not have to match. A *Riemann sum* S_g of a function $g(x)$ is defined as:

$$S_g = \sum_{i=1}^{u} g(x_i)\,\Delta x_i \tag{B.4}$$

where $\Delta x_i = x_i - x_{i-1}$ and i is the index, such as Measured Depth. u is the quantity of equidistant x-coordinates used to calculate the Riemann sum. A

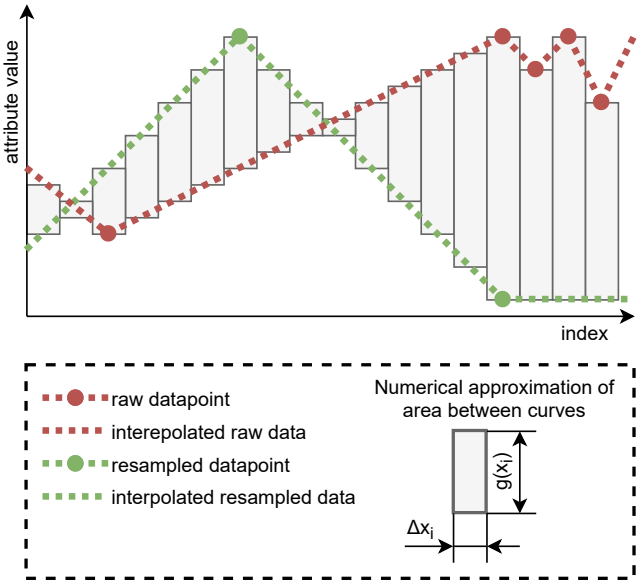


Figure B.16: Numerical difference visualization

square function can be easily added to the distance element of the sum, $g(x_i)$, making it a sum of difference-squared, or

$$SS_g = \sum_{i=1}^{u} \left(g(x_i)\right)^2 \Delta x_i. \tag{B.5}$$

Additional modification is necessary, since the integral, or the total sum, which indicates the total cumulative error may not be particularly informative. The average error is more intuitive, as it is independent of the actual length of the dataset, hence the equation can be further transformed to a mean squared error (MSE):

$$MSE_g = \frac{1}{u} \sum_{i=1}^{u} \left(g(x_i)\right)^2. \tag{B.6}$$

It can be done under an assumption that the Δx_i is constant. Finally, the equation can be transformed to a Root Mean Square (RMS) equivalent, here denoted as Root Mean Riemann Squared (RMRS):

$$RMRS_g = \sqrt{\frac{1}{u}\sum_{i=1}^{u}\left(g(x_i)\right)^2}. \tag{B.7}$$

Remark 2 For the Riemann sum to correctly approximate the value of the integral, relatively high resolution of calculation has to be employed (here: high value of u). Since the $r(\cdot)$, $t(\cdot)$, and consequently $g(\cdot)$ are defined as a tabulated data with linear interpolation between the consecutive points, the quantity u has to be higher than the quantity of datapoints in the resampled dataset. Results in this paper were calculated at u being 10 times higher than the quantity of resampled datapoints. The higher the value the closer the approximated function will be to the true value, approaching asymptotically.

To validate the method, it can be applied to the previously used synthetic dataset, where results were shown in Figure B.15. Error calculated using RMRS matches, for all intents and purposes, the calculations using the ground truth nearly perfectly, as seen in Figure B.17 which is nearly identical to Figure B.15, with R^2 correlation coefficient between the two being between 0.95 and 0.997.

In Appendix, additional results for a synthetic function (a triangular function) are presented in Figures B.26, B.27, and B.28 with more modest results of R^2 between 0.945 and 0.997.

B.4.3 Resampling: Case Study

B.4.3.1 Resampling quality evaluation on drilling data

To visualize the difference between the raw and resampled data in practice Figure B.18 and Figure B.19 were produced, where a small section of the ROP data from the well F9d of Volve dataset is shown. This shows how detrimental to signal quality resampling can be. Radius Neighbor Regressor and K-Neares Neighbor algorithms were applied at three different configurations of maximum depth-step multipliers of 1, 20, and 100 as well as of neighbour count of 1, 20

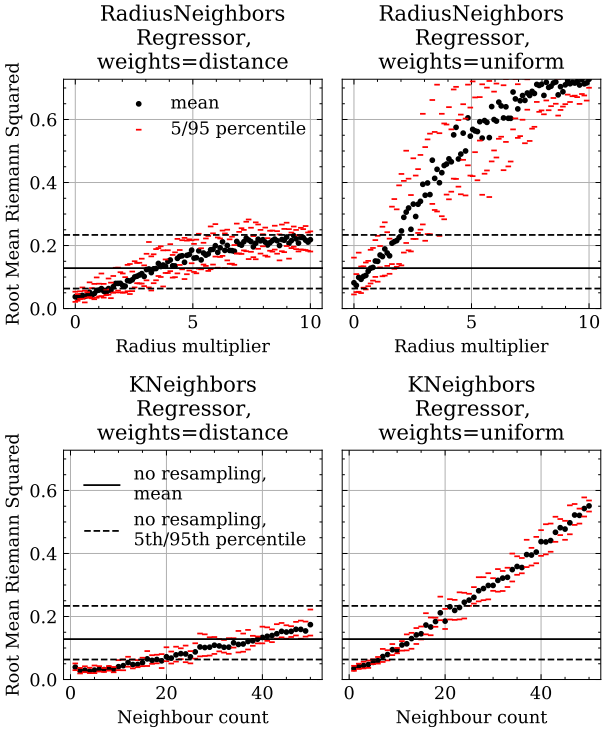


Figure B.17: Resampling error study

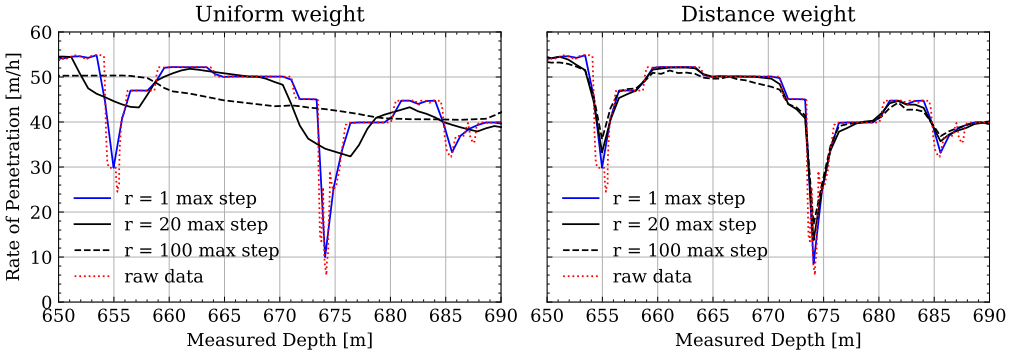


Figure B.18: Radius Neighbor Regressor, effects of different regressor radius on resampled data

and 100. When the uniform weight is used then even the relatively small radius distorts the signal.

In Figure B.18, inspecting signal for resampling using distance-based weights it

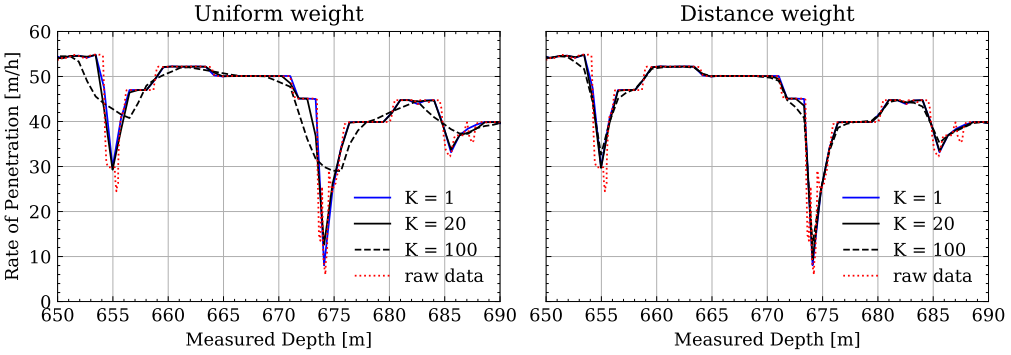


Figure B.19: K-Nearest Neighbors, effects of different neighbor count on resampled data

is clear that even at very high resampling radius the signal is effectively retaining all of its features. This makes this configuration most desirable. The key to this behaviour is the weight function applied to datapoints, which is the inverted distance, $w = 1/d$. In practice it means that points that are relatively far, even if they are within specified radius, will have a low weight attached and will not distort data significantly. To avoid having to manually inspect the data after resampling the RMRS method of calculating the deviation between original and resampled data is employed.

A sample result of the RMRS calculation based on Equation (B.7) is shown in Figure B.20. Here, a Stand Pipe Pressure (SPP) attribute from the well F9a of the Volve dataset is evaluated for four different resampling alorithms, Radius Neighbour Regressor and K-Nearest Neighbour with both uniform and distance based weights. Different radiuses and neighbour quantity is utilized. Corroborating manual inspection findings, the distance-weighted algorithms generally provide a lower error. As expected, the higher the radius and higher the neighbour quantity the higher the error, since the signal is being smoothed out.

Remark 3 It is worth noting that the error discussed here indicates the difference between the resampled and original data and is not indicative of the expected error for the further applied model. It does however indicate which resampling method keeps the resampled data closest to the truth, and therefore being desirable for

use. It may be the case that the signal is noisy and therefore replicating the noise in the resampled data is undesirable. However in such case it is worth considering denoising the resampled data as a separate step, before or after resampling, to keep better control over the process.

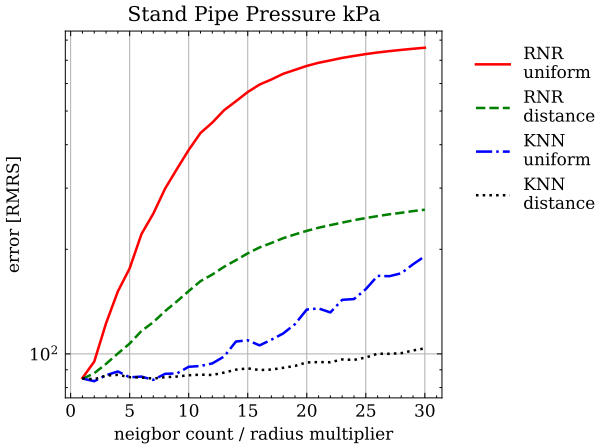


Figure B.20: Evaluation using RMRS example

Using the proposed RMRS method it is possible to evaluate which algorithms combination is the best for all the attributes in an example well. This exercise was performed on the well F9a, rows 2000 - 10000 (longest clean stride), of the Volve dataset and the results are presented in the Table B.2, where 112 attributes are used for resampling quality evaluation with different algorithms. It is clear that the majority of the attributes can be best resampled using Radius Neighbor Regressor, uniform weight with radius equal to 1 or 2 times maximum index step of the raw data. This, however, seems to be counter-intuitive when compared with Figure B.20, where this algorithm shows rapid error increase with K and radius.

For further insights additional analysis was performed where average performance of different algorithms was compared to the best algorithm for a given attribute. In other words, how much the RMRS error would increase if one was to always choose a fixed algorithm "blindly" compared to evaluating all possibilities and selecting the best one. Results are shown in Figure B.21. These results show that if one resampling algorithm was to be used for all algorithms "*blindly*" it is best

Table B.2: Algorithms that provide the lowest RMRS, out of 112 attributes

Algorithm	N / radius	Winner count
RNR Uniform	1	49
RNR Uniform	2	15
RNR Distance	2	9
RNR Uniform	3	4
RNR Uniform	4	4
RNR Distance	3	4
RNR Distance	1	3
KNN Uniform	7	3
other	n/a	21

to either use Radius Neighbour Regressor, distance weighted, with radius of 2 times the biggest original index step, or K-Nearest Neighbour, distance weighted, with 15 neighbors. Either of those two options come with penalty of under 5 percent of additional error compared to best algorithm. Out of those two the distance weighted KNN is likely a better choice, as it overall provides low error no matter the exact neighbour count selected.

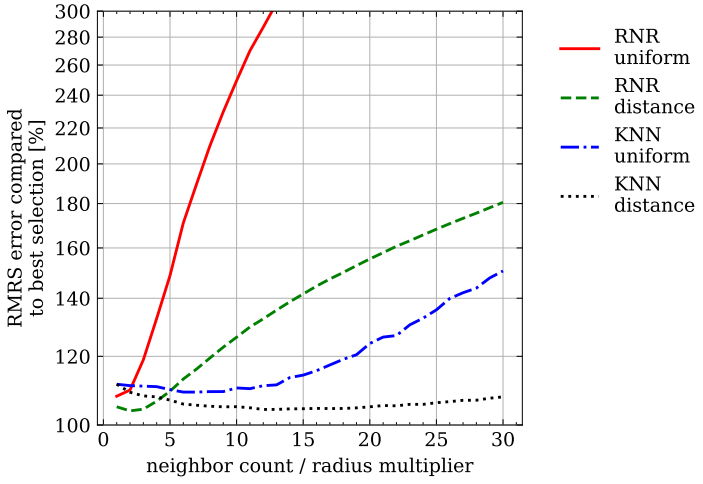


Figure B.21: Comparing a fixed algorithm throughout the dataset versus always picking best one.

B.4.3.2 Resampling method selection effect on prediction quality

Considering the findings from the previous sections, it is worthwhile to evaluate the influence of the re-sampling methods on the overall performance of machine learning model. Additionally it is possible to benchmark against non-resampled signals. To achieve an apples-to-apples comparison, the signal has to be downsampled so that the quantity of predicted time-steps as well as the average prediction horizon is the same as for other methods. This makes it possible to use identical neural network structure for all cases. This is achieved in practice by skipping a fixed number of rows in gap-filled dataset such that the quantity, and therefore the mean step size is identical to the case when data is re-sampled at specific, selected frequency.

Benchmarking was performed via complete *training while drilling* scenario when continuous inclination is predicted ahead of the bit while bent-sub drilling, as described in earlier publication [10], where the training dataset is continuous section from *0%* to *x%* of well drilled, while testing dataset is from *x%* to *(x+20)%* of the well drilled, ref. previously shown Figure B.7. There are multiple iterations for the model training and evaluation for $x \in (15, 80)$ in 50 evenly spaced percentage-steps. The sampling rate is set at five times the mean increment of the index value (measured depth), which results in approximately 100 data-rows for 25 meters of data.

Note that here it is the re-sampling algorithm settings that are evaluated, not the sampling rate. Effect of the sampling rate selection is explored in the next subsection. Results are evaluated as the MAE of the prediction. Data was taken from well F9A, depth based dataset, from Equinor's Volve data [2] converted to CSV files [3]. This dataset contains all the files related to a Volve field that was in operation between 2008 and 2018, including seismic, drilling, and production data.

Figure B.22 shows the results in terms of the MAE as a function of radius that is described by a multiplier for maximum index step, which in this case study

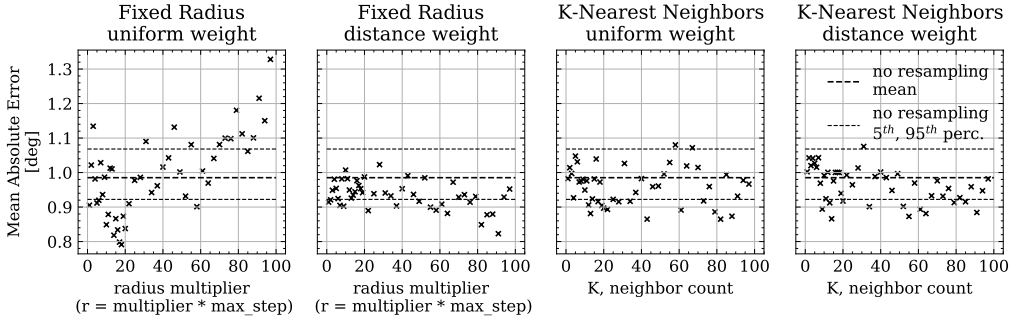


Figure B.22: Resampling error study, results in terms of mean absolute error

is 0.15 meters. A multiplier for maximum step is used to ensure that radius regressor will find at least one data point in the series for any arbitrary location.

Two algorithms were evaluated, K-Nearest-Neighbors and Fixed Radius Regressor, and two datapoint weighing strategies, uniform and distance based. Additionally a line is produced on each chart indicating mean, 5^{th}, and 95^{th} percentiles, results where data was not re-sampled, but downsampled (based on 100 runs at the same settings to calculate mean and percentiles). The results indicate that the radius equal to approximately 18 maximum index derivatives, or maximum steps, for Fixed Radius Regressor at uniform weight showed best results with MAE at 0.8 degrees, although the results are fairly spread out.

There are also additional practical reasons for choosing a low radius for Fixed Radius Regressor as well as low K for K-Nearest Neighbors algorithm. From practical point of view, if a radius of 5 meters is selected, then predictions will have to be delayed until further 5 meters of data is available; a resampled datapoint at MD=x takes into account points between MD=x-5 [m] and MD=x+5 [m]. This may or may not be an issue, depending on the specific application.

B.4.3.3 Sampling rate selection effect on prediction quality

When re-sampling or down-sampling the dataset, the new constant sampling rate has to be decided. In the process presented in this paper, the sampling rate is based on the mean index increments. To evaluate practical effects of the

sampling rate, again a Monte Carlo simulation was employed using the continuous inclination prediction scenario as described earlier.

Results of the inclination prediction study are presented in Figure B.23. A clear trend is visible, where the longer the sampling rate results in the higher mean absolute error. While the results are noisy, there is a visible area for the shortest sampling rates, where the error suddenly increases defying the general trend. This is most likely an artifact of the specific neural network size used. Hyperparameter tuning was done only once for the re-sampling rate equal to five times the mean original sampling rate, which is the minimum in the results. It is prohibitively time-expensive to perform hyperparameter tuning for each inspected re-sampling rate. It is likely that network layer size, dropout rates and learning rates can be adjusted to prevent that error increase, and therefore a high (short) sampling rate should be preferred, although that is achieved at the cost of the computational power needed.

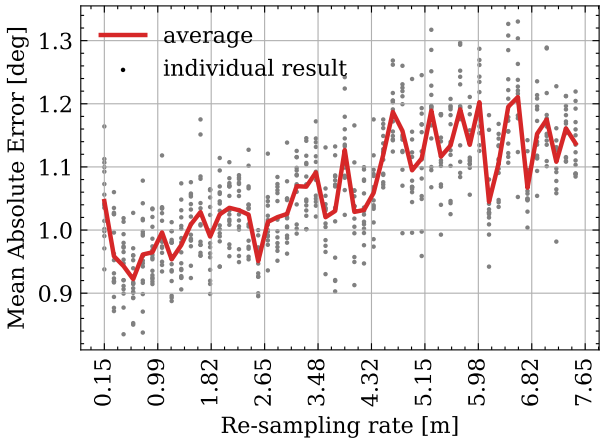


Figure B.23: Sampling rate study

B.5 Attribute selection and PCA configuration

Two attribute selection strategies were implemented in the semi-automatic process presented in this paper, Pearson correlation coefficient [24], and Predictive Power

Score (ppscore) [25] which provides asymmetric correlation coefficients based on decision trees. As an alternative to attribute selection PCA transformation is also available. Two scaling methods were tested as a pre-processing before applying PCA: normalization (ie. *(0,1)* scaling) and standardization (scaling so that standard deviation becomes 1).

Simulation was set up to evaluate the mean absolute error for all four approaches to reducing the quantity of inputs, as well as the actual quantity of inputs selected. Again, prediction of continuous inclination case study was chosen for this purpose. Figure B.24 shows the results, where PCA approach provides significantly lower error, without any difference between scaling strategies. This result is understandable considering the data; while continuous inclination is predicted, there is no direct correlation between the details of the inclination signal and drilling parameters, as the inclination is constantly rising in the analyzed curved section with varying rate. This is likely to be also the case for problems where recurrent neural networks perform the best. This shows that PCA transformation is a very convenient tool for input quantity reduction, as it captures the behaviour of the dataset independent of the target attribute. Pearson algorithm, at 80% of available well data, selected following attributes as correlated with inclination: *Hole depth (MD) m*, *Bit Drill Time h*, *Hole Depth (TVD) m*, *Extrapolated Hole TVD m*, and *Corrected Total Hookload kkgf*. While those values are technically corrected with inclination data, they are not particularly useful for a high quality prediction. ppscore algorithm picked similar attributes: *Extrapolated Hole TVD m*, *Pump Time h*, *Measured Depth m*, *Total Bit Revolutions unitles*, and *Hole depth (MD) m*.

An alternative simulation was performed, where the same case study was explored with the exception that the inclination values were converted to inclination change before feeding it into the model. The transformation was reversed to nominal inclination after the model's output. Such process is related to feature engineering, where attributes are altered to make the prediction process easier for the machine learning algorithm. There are pros and cons for such methods, however this is outside of the scope of this article.

This change allows the pearson correlation and ppscore algorithm to work on par with PCA, as seen in Figure B.25. In this case all algorithms perform similarly, although the spread in the results is significant. This is again a regrettable artifact of hyperparameters setup for slightly different problem (nominal inclination) and the quality can be improved by adjusting the neural network layer size, dropout rates and learning rate. Best score was again achieved when inputs were converted via PCA, this time input quantity of 3 showed the best results.

When investigating attributes selected is becomes clear that converting the target to inclination change allowed the algorithms to work as intended. Much more reasonable selection was done using pearson algorithm, yielding: *Total Hookload kkgf,* A*verage Rotary Speed rpm*, *Total Downhole RPM rpm*, *Average Surface Torque kN.m*, and *Weight on Bit kkgf.* ppscore selection was less obvious, listing *Total Downhole RPM rpm*, *Average Rotary Speed rpm*, M*WD DNI Temperature degC*, *Extrapolated Hole TVD m*, and *Corrected Total Hookload kkgf.*

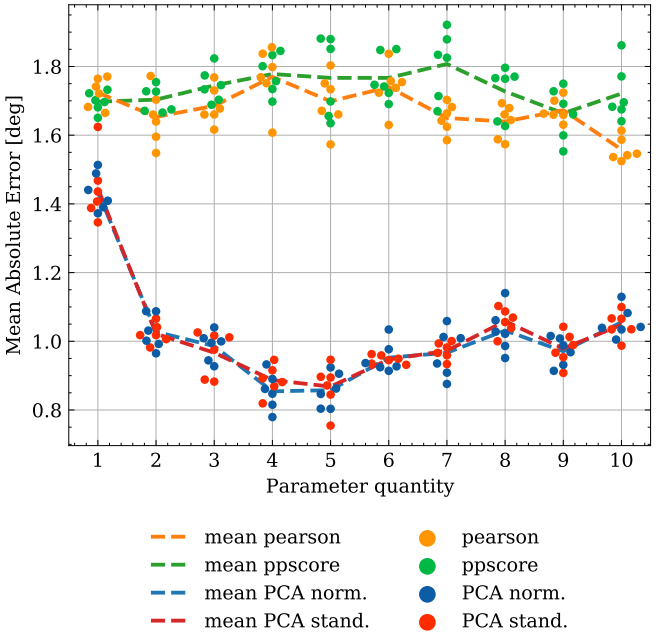


Figure B.24: Model selection strategy for MWD Continuous Inclination prediction

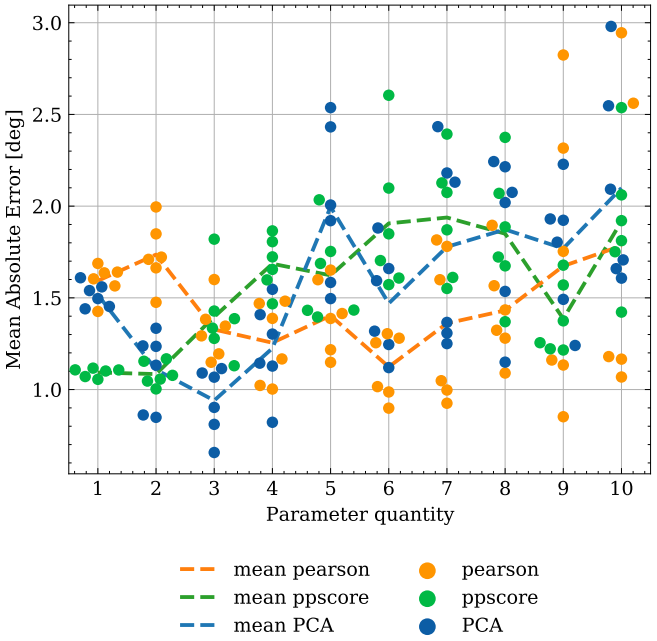


Figure B.25: Model selection strategy for MWD Continuous Inclination prediction based on inclination change

Remark 4 The main takeaway here is that it is unwise to depend on one "best" correlation algorithm to select inputs, as simple change to the case study makes the results very different. Additionally, in an automated approach where attributes are selected without a manual expert selection, PCA is a good alternative to typical correlation methods that will may when a more complex algorithm, such as RNN is used.

Two examples of correlation plots are reproduced in the appendix, Figure B.29 for Pearson coefficient and Figure B.30 for ppscore. It is worth noting that ppscore, unlike pearson correlation, does not generate symmetric results, which is rooted in the way how the algorithm works.

B.6 On Neural Network architectures

Work presented in this paper utilizes RNN for prediction purposes, namely

GRU; this also makes it is trivial to substitute it with vanilla RNN or LSTM. Considering the rapid pace of research within ML, it is necessary to point out that alternative architectures exist, both newer and older. Architecture selection can make a model less or more affected by the pre-processing, such as methods presented in this paper, and therefore further work is needed to understand this problem. This paper utilized RNNs in a case study where there was known benefit from including temporal information in the model [10], and therefore various approaches to pre-processing were likely to affect the results.

B.6.1 One to One, One to Many, Many to One, Many to Many

There are four distinct approaches to time series prediction in terms out input-output configuration. Input can consist of one single row of the data-series, i.e. X_i, or many rows, f.ex. $[X_{i-2}, X_{i-1}, X_i]$. Outputs can also be analogous, where only one step is predicted at a time, i.e. $\hat{x}_{i+1}$, or many steps, f.ex. $[\hat{x}_{i+1}, \hat{x}_{i+2}, \hat{x}_{i+3}]$. These variants can therefore be made into four distinct configurations, commonly referred to as *One to One*, *One to Many*, *Many to One*, and *Many to Many*.

Using just one row for input has a limitation that very limited temporal information can be learned by the network, limited in practice to predicting the first derivative behaviour.

In terms of outputs it is possible to predict multiple steps when predicting just one step by reusing previous predictions, where predicted value $\hat{x}_{i+1}$ can be used as an input, allowing prediction of $\hat{x}_{i+2}$ using the same architecture. This paper utilizes Many to Many approach, where multiple steps are the outputs of the ML model, as visualized in Figure B.5, as this was the approach that performed better, however no extensive research was done to benchmark the alternatives against each other.

Whether it is better to predict one or many will highly depend on the case study at hand. In the problem of inclination prediction explored in this paper predicting *many* was selected because the inclination will generally rise in the curved portion

of the well. This will cause error to accumulate if predict *one* approach was to be selected due to re-use of outputs needed to predict further values. In other case studies this may not be the case and predicting one value at a time might be the better solution.

B.6.2 Transformers

In 2017 a new architecture was introduced called a Transformer [40]. It displaced LSTM as the state of the art for natural language processing (NLP) by using attention mechanism. This allows the network to keep *focus* on the elements further down the sequence of inputs compared to RNNs. While it is not a problem fully analogous to drilling it does indicate that this new architecture should also be applicable for such problems. At the same time it is not given that more complex approach will perform better. There is a paper published in Nature [41] that put a single neuron against an earlier published deep learning network for a problem of predicting earthquake aftershocks matching the results. Further work is needed to establish when it is desirable to apply Transformer and when RNN in drilling.

B.6.3 Convolutional Neural Networks (CNN

Another potential approach is to use CNN architecture, which was successfully applied in drilling problems before to determine rock strength parameters [42] or lithofacies recognition [43]. While CNN are typically applied to classification problems, as opposed to a regression problem presented in this paper, a CNN element introduction either within the network itself, or as a separate step identifying sliding and rotating portions of drilling may bring benefits, and even make the proposed architecture more universal.

B.6.4 Linear regression, decision trees, support vector machines

As indicated in relation to Transformers, it is not given that new, more complex methods bring improved performance. This at the same time means that it is possible that simpler methods can perform better than the architecture presented here. For example decision trees were successfully used to fill in data gaps in drilling data [44]. While the case study of predicting inclination was explored in the past to benchmark performance against simpler methods [10] showing big benefits of RNNs, it is not guarantied to be the case for all problems.

B.7 Conclusion

A number of conclusions can be drawn from the performed studies related to optimal setup of machine learning processes in real-time drilling related problems.

1. Pre-processing of data plays a significant role in the quality of prediction using Recurrent Neural Networks
2. There is a significant difference in results depending whether small data gaps are filled using forward filling, linear interpolation, or using a smart selection of the two.
3. Re-sampling using fixed radius regressor with a small radius and uniform weight provided lowest prediction error, however it results in significant data degradation, and re-sampling with distance based weights provided similar results without the high potential for data degradation seen from the uniform weight approach. Evaluating the difference between raw and resampled data using proposed method of Root Mean Riemann sum Squared is an efficient way of gauging the quality of resampling.
4. Re-sampling rate has significant effect on accuracy, while benefiting from much faster computation time

5. Performance of attribute selection strategies can vary significantly with PCA being a good alternative to commonly used Pearson correlation coefficient
6. Predictive Power Score (ppscore) algorithm, although has significant theoretical improvements over the Pearson coefficient did not provide practical improvements in evaluated case study.

We hope that this paper highlights the need of proper data preparation in terms of gap filling and resampling, for research in both petroleum and other fields.

B.7.1 Future work

Future work is needed to better understand some effects seen in the presented research. This paper identified that there are differences in models' performance based on gap filling strategies, the root cause of this effect remains elusive. Additional case studies, RNN and non-RNN, would be beneficial to better quantify the effects of the re-sampling rate, especially utilizing hyperparameter tuning for different settings, which is extremely demanding of computational resources. More research towards the behaviour of the RNN models is necessary to understand better how the length of the input, length of the output, model update frequency, etc. influence the results.

B.8 Appendix

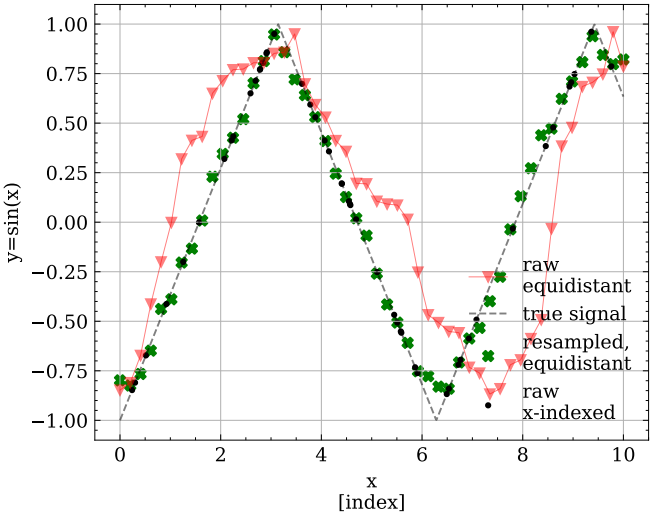


Figure B.26: Example of resampling issues, triangular function

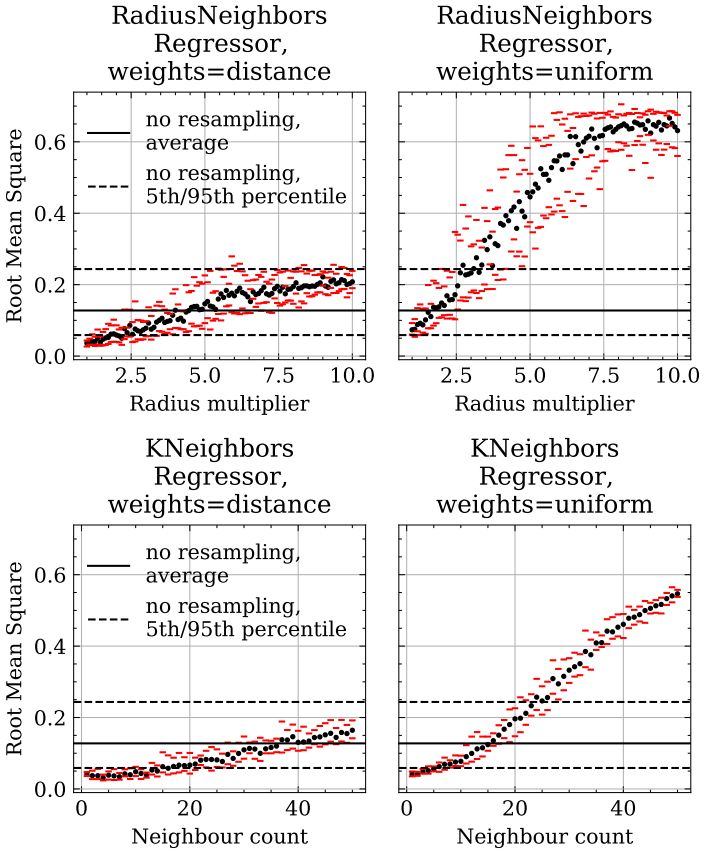


Figure B.27: RMS error between resampled values and ground truth

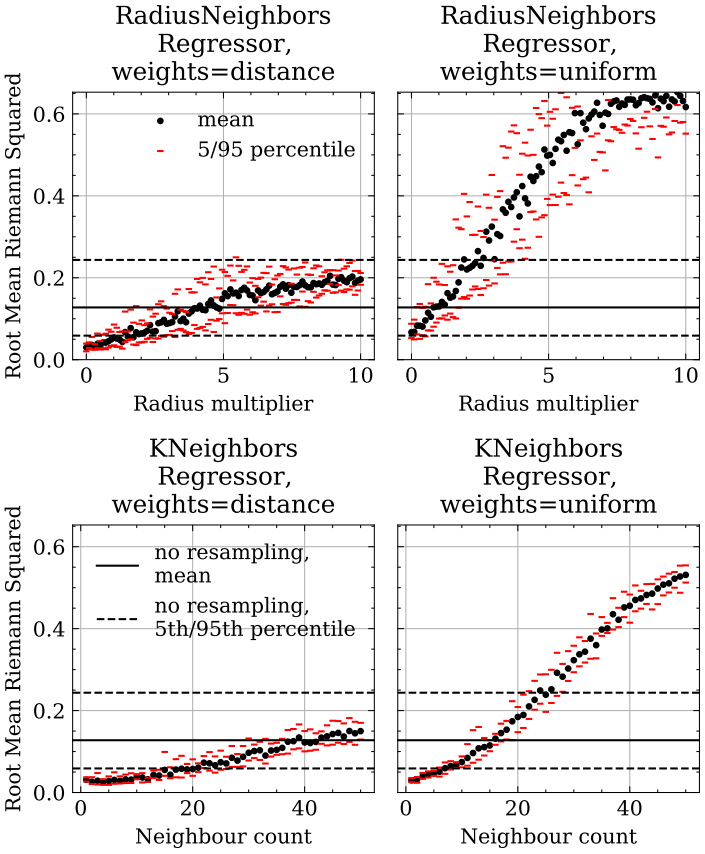


Figure B.28: RMS as calculated through RMRS

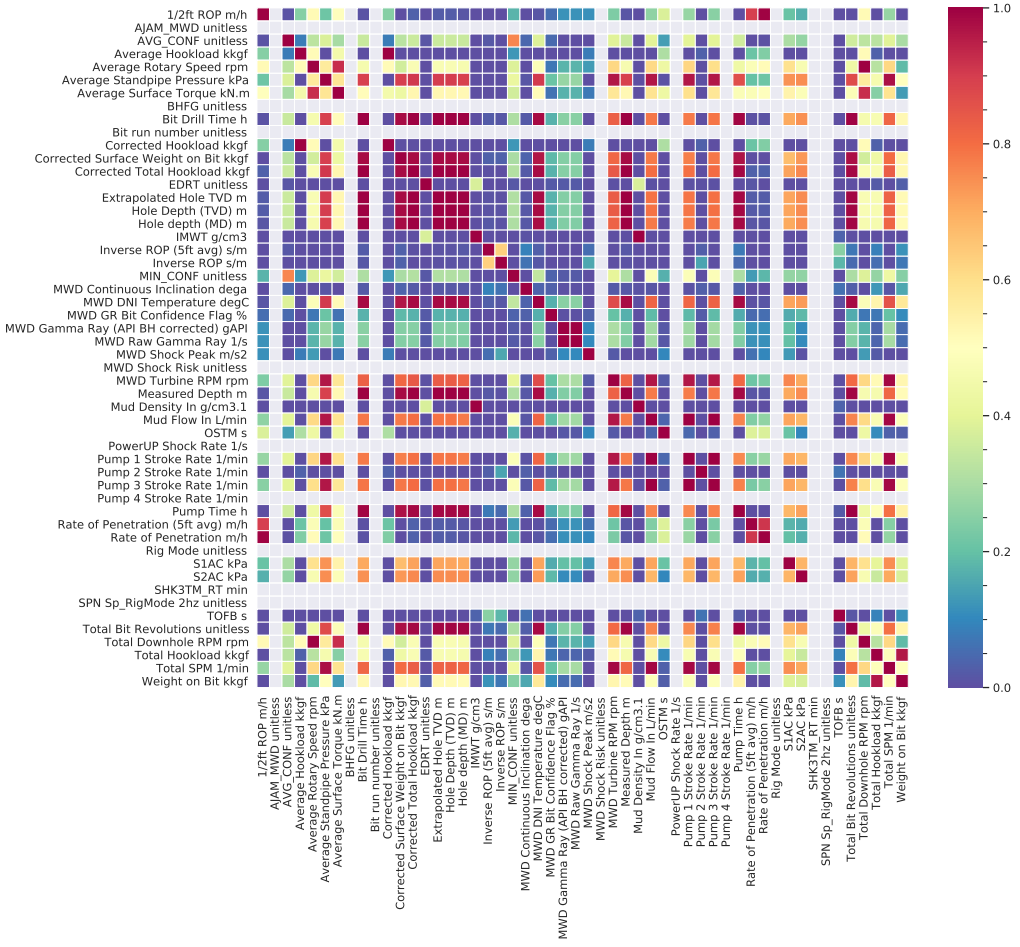


Figure B.29: Volve well F9a, Pearson correlation

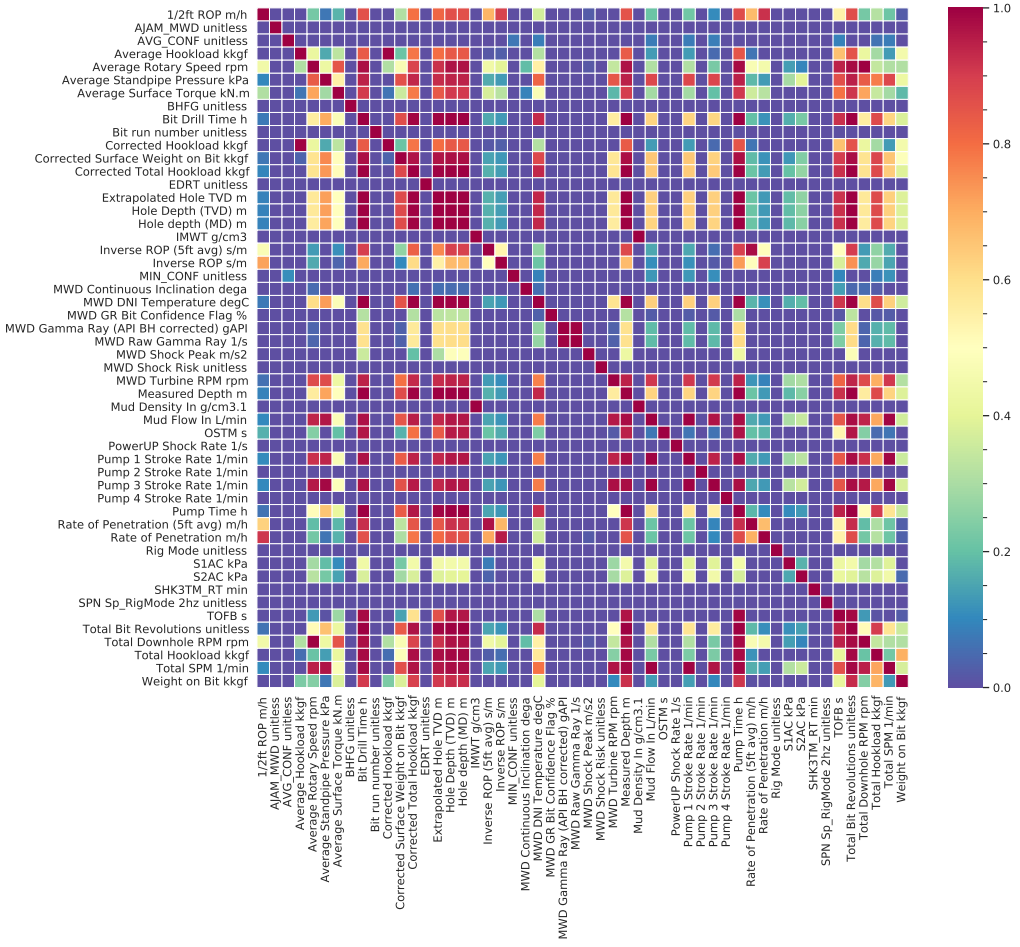


Figure B.30: Volve well F9a, ppscore correlation

Bibliography

[1] Luís Felipe F.M. Barbosa, Andreas Nascimento, Mauro Hugo Mathias, and João Andrade de Carvalho. "Machine Learning Methods Applied to Drilling Rate of Penetration Prediction and Optimization - A Review". In: *Journal of Petroleum Science and Engineering* 183 (Dec. 2019), p. 106332. ISSN: 09204105. DOI: 10.1016/j.petrol.2019.106332.

[2] Equinor. *Volve Field Data (CC BY-NC-SA 4.0)*. 2018. URL: https://www.equinor.com/en/news/14jun2018-disclosing-volve-data.html.

[3] Andrzej T Tunkiel, Tomasz Wiktorski, and Dan Sui. "Drilling Dataset Exploration, Processing and Interpretation Using Volve Field Data". In: *ASME 2020 39th International Conference on Ocean, Offshore and Arctic Engineering* (2020). DOI: 10.1115/OMAE2020-18151.

[4] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Representations by Back-Propagating Errors". In: *nature* 323.6088 (1986), pp. 533–536. ISSN: 00280836. DOI: 10.1038/323533a0.

[5] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures". In: *Neural Computation* 31.7 (2019), pp. 1235–1270. DOI: 10.1162/neco\{_\}a\{_\}01199. URL: https://doi.org/10.1162/neco_a_01199.

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3 (2014). URL: http://arxiv.org/abs/1412.3555.

[7] Suihong Song, Jiagen Hou, Luxing Dou, Zezhang Song, and Shuang Sun. "Geologist-Level Wireline Log Shape Identification with Recurrent Neural Networks". In: *Computers and Geosciences* 134 (Jan. 2020), p. 104313. ISSN: 00983004. DOI: 10.1016/j.cageo.2019.104313.

[8] Khim Chhantyal, Minh Hoang, Håkon Viumdal, and Saba Mylvaganam. "Flow Rate Estimation Using Dynamic Artificial Neural Networks with Ultrasonic Level Measurements". In: *Proceedings of the 9th EUROSIM Congress on Modelling and Simulation, EUROSIM 2016, the 57th SIMS Conference on Simulation and Modelling SIMS 2016*. 142. Linköping University Electronic Press. 2018, pp. 561–567.

[9] Augustine Osarogiagbon, Somadina Muojeke, Ramachandran Venkatesan, Faisal Khan, and Paul Gillard. "A New Methodology for Kick Detection during Petroleum Drilling Using Long Short-Term Memory Recurrent Neural Network". In: *Process Safety and Environmental Protection* (2020).

[10] Andrzej T Tunkiel, Dan Sui, and Tomasz Wiktorski. "Training-While-Drilling Approach to Inclination Prediction in Directional Drilling Utilizing Recurrent Neural Networks". In: *Journal of Petroleum Science and Engineering* 196 (2021).

[11] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzel. *Learning to Diagnose with LSTM Recurrent Neural Networks*. 2015. arXiv: 1511.03677.

[12] Andrzej T Tunkiel, Tomasz Wiktorski, and Dan Sui. "Continuous Drilling Sensor Data Reconstruction and Prediction via Recurrent Neural Networks". In: *ASME 2020 39th International Conference on Ocean, Offshore and Arctic Engineering*. 2020. DOI: 10.1115/OMAE2020-18154.

[13] Andrzej Tunkiel. *Github for TOPPMEIS Project*. URL: https://github.com/AndrzejTunkiel/Tape.

[14] Andrzej T. Tunkiel, Dan Sui, and Tomasz Wiktorski. "Data-Driven Sensitivity Analysis of Complex Machine Learning Models: A Case Study of Directional Drilling". In: *Journal of Petroleum Science and Engineering* 195 (Dec. 2020), p. 107630. ISSN: 09204105. DOI: 10.1016/j.petrol.2020.107630.

[15] Dan Sui, Olha Sukhoboka, and Bernt Sigve Aadnøy. "Improvement of Wired Drill Pipe Data Quality via Data Validation and Reconciliation". In: *International Journal of Automation and Computing* 15.5 (Oct. 2018), pp. 625–636. ISSN: 17518520. DOI: 10.1007/s11633-017-1068-9. URL: http://link.springer.com/10.1007/s11633-017-1068-9.

[16] Suranga C.H. Geekiyanage, Dan Sui, and Bernt S. Aadnoy. "Drilling Data Quality Management: Case Study with a Laboratory Scale Drilling Rig". In: *Proceedings of the International Conference on Offshore Mechanics and Arctic Engineering - OMAE*. Vol. 8. American Society of Mechanical Engineers, June 2018, pp. -. ISBN: 978-0-7918-5129-6. DOI: 10.1115/OMAE2018-77510.

[17] Suranga C H Geekiyanage, Andrzej Tunkiel, and Dan Sui. "Drilling Data Quality Improvement and Information Extraction with Case Studies". In: *Journal of Petroleum Exploration and Production Technology* (2020). ISSN: 2190-0566. DOI: 10.1007/s13202-020-01024-x.

[18] Andrzej T Tunkiel, Dan Sui, and Tomasz Wiktorski. "Reference Dataset for Rate of Penetration Benchmarking". In: *Journal of Petroleum Science and Engineering* (2020). ISSN: 0920-4105. DOI: 10.1016/j.petrol.2020.108069.

[19] Shachar Kaufman, Saharon Rosset, and Claudia Perlich. "Leakage in Data Mining: Formulation, Detection, and Avoidance". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, New York, USA: ACM Press, 2011, pp. 556–

563. ISBN: 978-1-4503-0813-7. DOI: 10.1145/2020408.2020496. URL: http://dl.acm.org/citation.cfm?doid=2020408.2020496.

[20] Shay Geller. *Normalization vs Standardization — Quantitative Analysis | by Shay Geller | towards Data Science*. 2019. URL: https://towardsdatascience.com/normalization-vs-standardization-quantitative-analysis-a91e8a79cebf.

[21] Karl Pearson. "LIII. On Lines and Planes of Closest Fit to Systems of Points in Space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572. DOI: 10.1080/14786440109462720.

[22] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. "Scikit-Learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[23] Wolfram Stacklies, Henning Redestig, Matthias Scholz, Dirk Walther, and Joachim Selbig. "pcaMethods—a Bioconductor Package Providing PCA Methods for Incomplete Data". In: *Bioinformatics (Oxford, England)* 23.9 (May 2007), pp. 1164–1167. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btm069. URL: https://doi.org/10.1093/bioinformatics/btm069.

[24] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. "Pearson Correlation Coefficient". In: *Noise Reduction in Speech Processing*. Springer, 2009, pp. 1–4.

[25] Florian Wetschoreck, Tobias Krabel, and Surya Krishnamurthy. "8080labs/Ppscore: Zenodo Release". In: (Oct. 2020). DOI: 10.5281/ZENODO.4091345. URL: https://zenodo.org/record/4091345.

[26] Antonello Pasini. "Artificial Neural Networks for Small Dataset Analysis". In: *Journal of Thoracic Disease* 7.5 (2015), pp. 953–960. ISSN: 20776624. DOI: 10.3978/j.issn.2072-1439.2015.04.61. URL: /pmc/articles/PMC4454870/%20/pmc/articles/PMC4454870/?report=

abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4454870/.

[27] Soheil Esmaeilzadeh, Amir Salehi, Gill Hetz, Feyisayo Olalotiti-lawal, Hamed Darabi, and David Castineira. "Multiscale Modeling of Compartmentalized Reservoirs Using a Hybrid Clustering-Based Non-Local Approach". In: *Journal of Petroleum Science and Engineering* 184 (Jan. 2020), p. 106485. ISSN: 0920-4105. DOI: 10.1016/J.PETROL.2019.106485.

[28] Soheil Esmaeilzadeh, Amir Salehi, Gill Hetz, Feyisayo Olalotiti-lawal, Hamed Darabi, and David Castineira. "A General Spatio-Temporal Clustering-Based Non-Local Formulation for Multiscale Modeling of Compartmentalized Reservoirs". In: *SPE Western Regional Meeting Proceedings* 2019 (Apr. 2019). DOI: 10.2118/195329-MS.

[29] Frank Rosenblatt. *Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms*. Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[30] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 08997667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735.

[31] Lutz Prechelt. "Early Stopping - but When?" In: Springer, Berlin, Heidelberg, 1998, pp. 55–69. DOI: 10.1007/3-540-49430-8\{_\}3. URL: https://link.springer.com/chapter/10.1007/3-540-49430-8_3.

[32] Bing Liu. "Lifelong Machine Learning: A Paradigm for Continuous Learning". In: *Frontiers of Computer Science* 11.3 (2017), pp. 359–361. ISSN: 20952236. DOI: 10.1007/s11704-016-6903-6.

[33] Ilya M Sobol. "Sensitivity Estimates for Nonlinear Mathematical Models". In: *Mathematical modelling and computational experiments* 1.4 (1993), pp. 407–414.

[34] H J Landau. "Sampling, Data Transmission, and the Nyquist Rate". In: *Proceedings of the IEEE* 55.10 (1967), pp. 1701–1706.

[35] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. *Phased Lstm: Accelerating Recurrent Network Training for Long or Event-Based Sequences*. 2016. arXiv: 1610.09513.

[36] Evelyn Fix. *Discriminatory Analysis: Nonparametric Discrimination, Consistency Properties*. Vol. 1. USAF school of Aviation Medicine, 1985.

[37] Jon L Bentley. *Survey of Techniques for Fixed Radius near Neighbor Searching*. Stanford Linear Accelerator Center, Calif.(USA), 1975.

[38] Russel E. Caflisch. "Monte Carlo and Quasi-Monte Carlo Methods". In: *Acta Numerica* 7 (1998), pp. 1–49. ISSN: 14740508. DOI: 10.1017/S0962492900002804.

[39] Nicole Engelke and Vicki Sealey. "The Great Gorilla Jump: A Riemann Sum Investigation". In: *Proceedings of the 12th special interest group of the Mathematical Association of America on research in undergraduate mathematics education* (2009).

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems* 2017-December (June 2017), pp. 5999–6009. URL: https://arxiv.org/abs/1706.03762v5.

[41] Arnaud Mignan and Marco Broccardo. "One Neuron versus Deep Learning in Aftershock Prediction". In: *Nature 2019 574:7776* 574.7776 (Oct. 2019), E1–E3. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1582-8. URL: https://www.nature.com/articles/s41586-019-1582-8.

[42] Mingming He, Zhiqiang Zhang, Jie Ren, Jiuyang Huan, Guofeng Li, Yunsheng Chen, and Ning Li. "Deep Convolutional Neural Network for Fast Determination of the Rock Strength Parameters Using Drilling Data". In: *International Journal of Rock Mechanics and Mining Sciences*

123 (Nov. 2019), p. 104084. ISSN: 1365-1609. DOI: 10.1016/J.IJRMMS.2019.104084.

[43] Rafael Pires de Lima, Fnu Suriamin, Kurt J. Marfurt, and Matthew J. Pranter. "Convolutional Neural Networks as Aid in Core Lithofacies Classification". In: *http://www.seg.org/interpretation* 7.3 (May 2019), SF27–SF40. DOI: 10.1190/INT-2018-0245.1. URL: https://library.seg.org/doi/abs/10.1190/INT-2018-0245.1.

[44] Runhai Feng, Dario Grana, and Niels Balling. "Imputation of Missing Well Log Data by Random Forest and Its Uncertainty Analysis". In: *Computers & Geosciences* 152 (July 2021), p. 104763. ISSN: 0098-3004. DOI: 10.1016/J.CAGEO.2021.104763.

Appendix C

Reference Dataset for Rate of Penetration Benchmarking, AT2

Tunkiel, A. T., Sui, D., & Wiktorski, T. (2021). **Reference dataset for rate of penetration benchmarking**. Journal of Petroleum Science and Engineering, 196, 108069.

Abstract

In recent years, there were multiple papers published related to rate of penetration prediction using machine learning vastly outperforming analytical methods. There are models proposed reportedly achieving R^2 values as high as 0.996. Unfortunately, it is most often impossible to independently verify these claims as the input data is rarely accessible to others. To solve this problem, this paper presents a database derived from Equinor's public Volve dataset that will serve as a benchmark for rate of penetration prediction methods. By providing a partially processed dataset with unambiguous testing scenarios, scientists can

perform machine learning research on a level playing field. This in turn will both discourage publication of methods tested in a substandard manner as well as promote exploration of truly superior solutions. A set of seven wells with nearly 200,000 samples and twelve common attributes is proposed together with reference results from common machine learning algorithms. Data and relevant source code are published on the pages of University of Stavanger and GitHub.

C.1 Introduction

Research review related to machine learning (ML) models within petroleum is problematic. To fully evaluate a proposed method both data and exact description of the method are necessary, which is regrettably rare in the field. Data is often withheld on the grounds of confidentiality and there is little pressure to release source code behind presented methods. It leaves the scientific discourse susceptible to errors or cheating [1], where results might be artificially inflated. Most researchers consider that there is currently a reproducibility crisis in science [2], with 70% of polled scientists admit to trying and failing to reproduce experiments.

Introduction of a standardized real-time well log dataset has a capacity to transform research in data-driven methods related to drilling. It has high potential to spark a healthy competition between researchers striving for better performance. Well known datasets such as MNIST promoted competition, and facilitated research and knowledge sharing in the field of handwriting recognition. Additionally, having source data available makes paper authors accountable, as any dishonest practices will be easily discoverable when others attempt to reproduce the results.

Building upon Volve dataset [3], made public by Equinor in 2018 on a very permissive license[1], and previous data preparation work [4], this paper proposes a standardized dataset of seven wells containing twelve commonly logged attributes for ROP prediction purposes. In addition to data itself, three benchmarking scenarios are proposed, with specific metrics attached to them, ensuring that future

[1]Creative Commons BY-NC-SA 4.0, https://creativecommons.org/

results are comparable with each other. To establish a point of reference results from a number of basic algorithms are presented, together with complete source code[2] needed for result replication as well as a starting point for other researchers. We hope that it will enable reproducibility, competition, and cooperation between the researchers elevating the quality of papers published in the field. It also has potential of lowering the entry point for data-driven methods in drilling as well as attracting talent from outside of petroleum field.

C.2 Existing problems and potential pitfalls

C.2.1 Rationale behind rate of penetration prediction and methodology

Purpose and goal

Drilling of oil wells is very expensive. IHS reports day rates of semisubmersible, and jack-up rigs, as well as drillships [5] from mid 2017 up to present day. At the time of this writing[3] average rates vary between 40,000 to 300,000 USD per day of drilling, therefore increasing the ROP translates directly to savings for the operator.

Additional area of potential savings is optimization of the mechanical specific energy (MSE) to ROP ratio required to drill. While the energy itself is not a significant portion of the drilling cost, its excess has to be released as effects other than cutting of the rock, as vibrations and heat, leading to higher tool wear and increased likelyhood of failure. This in turn can extend the drilling time due to additional tripping operations to exchange equipment, or fishing operations to retrieve equipment lost in hole.

Assumptions

[2] https://github.com/AndrzejTunkiel/USROP
[3] August 2020

ROP prediction model is necessary for ROP optimization. This historically is done both though analytical as well as data-driven models. Both approaches require reference data, to either find model constants [6] or to train the machine learning model. The closer the reference drilling is to future drilling, the smaller expected error between reality and the model should become. This also means, that reported accuracy of a model is only applicable to drilling at the same level of similarity, in terms of equipment, lithology, procedures, depth, etc. Reference, or training, data, can be taken from neighbouring wells, or from the currently drilled well in continuous learning scenario [7], where a model is created *on the go*.

ROP optimization process

ROP optimization can be performed in multiple ways. Figure C.1 presents a basic outline of two general approaches to such an optimization process. ROP model is used to determine optimal drilling parameters, related to a chosen optimization problem, be it minimizing drilling time, minimizing MSE used to drill or others. This in turn is to set typical drilling inputs, such as drill bit RPM or weight on bit. The difference between a reference well and continuous learning method is indicated in subfigures a) and b). A reference well can be used, which is similar to the well drilled, a). This allows for model development to happen offline and is done once, and is in general simpler to deploy; a reference well is however required, which may or may not be available. Alternatively, the model can be created *on the go* in a continuous learning fashion, as shown in the subfigure b). This requires model development to happen while drilling, which is more difficult to implement due to computing equipment and skills necessary. Additional drawback is that the initial dataset is very small, and empty at the very start, which is detrimental to machine learning training process. Temporary model can be used, or a *warm-up* period exists where there is no model present. This is countered by the fact that the data is much more closely related to drilling at hand, improving the performance. In both approaches the ROP model is used to calculate the optimized drilling parameters that are then applied to the drilling controls related to parameters such as weight on bit, drill string RPM, mud flow

etc. Method choice would depend on data availability, and other local constraints. Hybrid methods are also possible, and best practices are yet to be established.

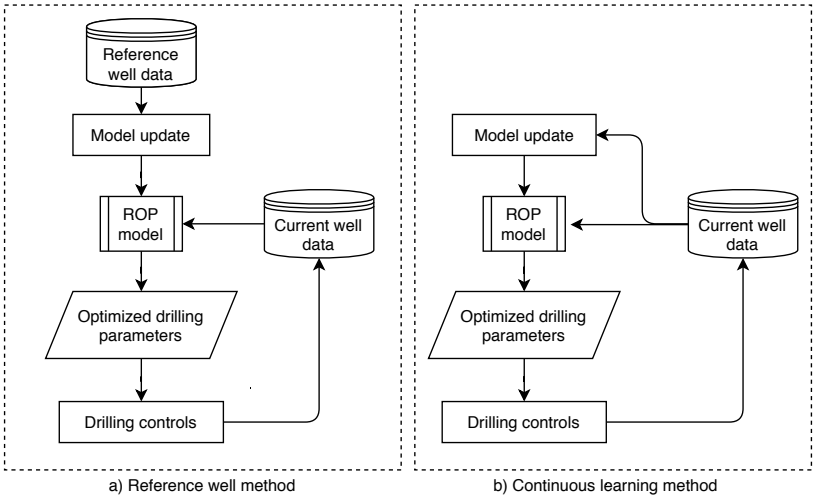


Figure C.1: ROP optimization method

C.2.2 Data availability

Initiating this research was a review of a number of recent papers [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25] that aimed at rate of penetration (ROP) prediction. Nearly none provided the data; we were able to identify few exceptions - two papers [21, 22] used data originally published in 1974, of which 30 samples [23] were directly quoted in the reference paper. Another identified paper provided 25 samples [24] that were subsequently re-used by others [25]. This means that the existing publications are either not reproducible or use a very small size dataset that does not meet the standards of modern machine learning research. For comparison, a popular open MNIST [26] dataset, a Modified National Institute of Standards and Technology database of handwritten digits contains 70,000 samples, which was later extended [27] to 300,000 samples. These datasets contributed to over 20,000 publications over 20 years since its inception.

C.2.3 Source code availability

Source code availability among reviewed papers was even poorer - none of the reviewed papers shared it. While mathematical description of the basic underlying method was common, it is rarely sufficient to reproduce results. Simple Multi-Layer Perceptron (MLP) in a popular Keras implementation has to be described at least by number of layers with number of neurons, activation of each layer, bias, kernel initializer, kernel regularizer, bias regularizes, activity regularizer, kernel constraint, bias constraint, training loss function, batch size, epoch count, an optimizer selection together with specific internal parameters such as learning rate, beta 1, beta 2, epsilon, and amsgrad status in case of the default adam optimizer. While most of those parameters have defaults, they differ between libraries and even version of the libraries themselves, making reproduction or result verification impossible if configuration information is incomplete. This problem can, and often is, solved via source code publication, often with a random seed fixed to achieve exactly the same results as published. A popular website *paperswithcode.com* tracks publications with source code available, as well as keeps leaderboards tracking performance improvement on selected datasets. This website has no drilling related papers listed. It is difficult to improve the state-of-the-art if it is not possible to reproduce it, making the goalposts invisible.

C.2.4 Incorrect data split

Sequential nature of real-time drilling data makes it susceptible to mistakes, for example related to train/test data split. One of the commonly logged attributes is measured depth, which with wrong data split, can inadvertently provide the model with information it should not have. Consider data where the only input attribute consists of evenly spaced values from 0 to 1. The target value is a random walk with step distance taken from normal distribution. There is absolutely no correlation between the input and the output. Yet if a random test/train split is applied, a common practice in ML, together with an off the shelf regression algorithm the resulting prediction R^2 score is 0.9948. This is because machine

learning algorithm will *learn* that for input 0.40 target is 55 and for input 0.42 target is 57, then, when asked for prediction for input 0.41 accurate answer is very easy (most likely about 56). Full implementation showing the problem in the Appendix as Listing C.1.

Even without the depth attribute there may be enough information for the model to correctly recognize that a certain sample is from the same area as the samples used for training and infer a correct output. Consider an absurd notion - Measured Depth prediction based on Surface Torque and Rotary Speed. While one can argue, that there would be some correlation, at least with the Surface Torque, these attributes are surely insufficient for an accurate prediction. Performing exactly this exercise, applying a random train/test split, it is possible to achieve impressive R^2 score of 0.946 for measured depth prediction using an off the shelf Gradient Boosting Regressor with default parameters - source code in Listing C.2. With automated best model selection, testing portion reduced to 10%, and a malicious random seed selection, R^2 of 0.998 was achieved, or 1.00 if rounded to two decimals.

Using random train/test split may be used if the goal of the model is to interpolate existing data or explore the relationships existing in specific well. This however has to be done with caution and with understanding that spurious correlations will be utilized by the model and it should not be used to predict values for other wells. For example, if drilling was unusually slow at Measured Depth of 1,040m and 1,050m, such model will correctly predict that it was also slow at MD of 1,045m. This however is not an indication that in a different well ROP will also be low at this specific depth, even though the model is likely to indicate that.

In Scikit-Learn [28] library an appropriate function exists for splitting sequence-type data; it is *sklearn.model_selection.TimeSeriesSplit*. While the name suggests that it is meant specifically for time series, it is also the correct tool to use for depth series type of data, or any sequential data where there is dependence between consecutive measurement points. This function divides the data into even, continuous splits. In the k^{th} split, it returns first k *folds* as train set and

the *(k+1)th fold* as test set. This creates multiple train/test sets in a structure suitable for continuous learning methods.

C.2.5 Easy target problem

Different problems can arise due to specific data selection and scoring. Consider data where the only input is a random number with average of 20 and standard deviation of 2. The target attribute is a random number with average of 50 and standard deviation of 3, so it ranges approximately between 40 and 60. No correlation exists between these two attributes, yet again, average error between the prediction and ground truth is only 4.8%. This is result of a very easy target, where simply guessing at the average, here 50, and ignoring the input will yield such an impressive result. The example code is shown in Listing C.3.

While all these described shortcomings do not necessarily mean that any of the published papers have flawed methodology or doctored results. It is nevertheless a fact that the bulk of published papers on data-driven ROP prediction is done on undisclosed data, using methods that are not described sufficiently for reproduction, and therefore present results that are impossible to verify. One cannot improve on the existing research without reproduction or at minimum - a presence of a unified benchmark.

C.3 University of Stavanger Rate of Penetration (USROP) dataset

C.3.1 Data source - Volve dataset

In 2018 Equinor published data related to the Volve field located off the coast of Norway. It was in operation in years 2008 - 2016 and in total produced 63,000,000 bbl of oil. The dataset contains data related to geoscience, production, seismic, reservoir modelling and drilling. What made the adoption of the data for research

relatively slow, is that the data is provided without any preprocessing that would make it more accessible. To facilitate research related to drilling, independent work was done to convert the real-time drilling logs from segmented WITSML files into compact CSV files [4]. This makes the data much easier to handle for the purpose of data science. The files are made available for download from the pages of University of Stavanger[4]. The Volve dataset is published on Creative Commons BY-NC-SA 4.0 licence, which allows anyone to modify and re-publish the data as long as the original source is attributed, it is done in a non-commercial fashion, and that the license is retained.

C.3.2 The data

The real-time drilling data from the Volve dataset is vast and allows for research in different sub-fields of drilling. At the same time the logs are of varying quality, containing missing data and different attributes. Significant clean up pre-work is required before feeding the data into ML algorithms, which can be to a high extent arbitrary. To solve this issue a curated subset of Volve is necessary. Based of analysis of the logged attributes total of seven wells were selected. The selection criteria was the completeness of the data and common attributes logged. Additionally only the depth-based real-time WITSML logs were used as opposed to time-based logs. Rationale behind this selection was that the time-based data would require additional processing which is currently outside of the scope of this study. The following wells were selected:

- Norway-NA-15_47_9-F-9 A depth
- Norway-Statoil-15_47_9-F-7 depth
- Norway-StatoilHydro-15_47_9-F-14 depth
- Norway-StatoilHydro-15_47_9-F-15 depth
- Norway-StatoilHydro-15_47_9-F-15S depth

[4]http://www.ux.uis.no/~atunkiel/

- Norway-StatoilHydro-15_47_9-F-5 depth
- Norway-StatoilHydro-15_47_9-F-9 depth

In terms of available attributes, the focus was on commonly logged data to promote models for wide application. Following attributes were selected: Measured Depth [m], Weight on Bit [kkgf], Average Standpipe Pressure [kPa], Average Surface Torque [kN.m], Rate of Penetration [m/h], Average Rotary Speed [rpm], Mud Flow In [L/min], Mud Density In [g/cm3], Diameter [mm], Average Hookload [kg], Hole Depth (TVD) [m], USROP Gamma [gAPI].

It is necessary to understand, that data in Volve, as well as generally in drilling rigs, is not collected in a standardized manner. Different equipment is used and operated using different procedures. This is often the reality of the oilfield operations and a method that is meant for future field deployment should be robust enough to overcome those challenges. Alternatively, all equipment could be properly and identically calibrated, as highlighted by [29]. This is an important distinction to note when comparing methods and results.

Minimal processing was done to the attributes to preserve original data. This is necessary as the drilling logs often contain erroneous, non-physical values. There may be sentinel values to indicate no reading (typically -999), corrupted values coming through the mud-pulsing system, and others. Samples containing *Weight on Bit* values below 0 and above 35 were truncated. The same way rows with *Mud Density In*, *Mud Flow In*, and *Average Surface Torque* values below zero were removed, as well as with *Rate of Penetration* values above 100 and *Average Standpipe Pressure* above 25,000. *Diameter* refers to the nominal wellbore diameter. Forward and backward filling was used to fill in the small gaps in the data resulting from uneven logging frequency of different equipment.

There was no unified gamma reading between all the wells, hence a new attribute, *USROP Gamma*, was introduced. It contains data logged under different names and different equipment, sometimes even within the same well. Source code is provided for exact explanation of which gamma related attribute was used in each

case. The dataset was balanced in terms of samples per measured depth available. The goal was to remove the variability in the polling rate, so that a given length of a well has the same weight in terms of error, when it is calculated as a total of multiple wells. Well with fewest samples per available depth was identified and sample count of other wells was reduced through random sampling to match the identified value. Additionally, it is typically beneficial for ML approaches to balance the dataset anyway. It prevents error minimization algorithms from being overwhelmed by an over-represented value. In simple terms, an algorithm differentiating between cats and dogs will be best trained when the dataset is split evenly between these two classes. If cats were to be over-represented in a ratio of 9:1, algorithm may settle on a local minimum where everything is a cat with 90% accuracy. Complete source code used for data preparation is made available on GitHub[5].

The described pre-processing results in data from seven wells with total available measure depth ranging from 332m to 2,759m. This translates to 6,389 and 53,041 data samples respectively with total of 198,928 samples and 10,346 meters of measured depth among all the wells. Table C.1 provides the exact breakdown of those values. The names of the well were changed so that they are easily identifiable, such as *USROP_A 2 N-SH_F-14d.csv. USROP* stands for ***U****niversity of* ***S****tavanger* ***R****ate of* ***P****enetration.* ***A*** refers to the revision of the dataset. ***2*** is a short well identifier, and ***N-SH_F-14d*** refers to original CSV file titled ***N****orway-****S****tatoil****H****ydro-15_47_9-****F-14 d****epth.csv.* For brevity, this paper henceforth will refer to the wells simply by their consecutive number, such as *well 2*.

To provide a general insight in how the quality of the data well 2 is reproduced in the Appendix as Figure C.13. This chart is for reference only to provide a general feel for the dataset for potential researchers. Some outliers are present, changes in equipment are visible both in terms of well diameter, as well as gamma reading, which abruptly changes.

Among the attributes available in the presented dataset one can notice that lithology information is not present. This was done because such data is not

[5]https://github.com/AndrzejTunkiel/USROP

Table C.1: USROP dataset well depth reference

Filename	Starting MD [m]	Final MD [m]	Available length [m]	Sample count
USROP_A 0 N-NA_F-9_Ad.csv	491	1,206	715	13,746
USROP_A 1 N-S_F-7d.csv	301	634	332	6,389
USROP_A 2 N-SH_F-14d.csv	988	3,466	2,478	47,645
USROP_A 3 N-SH-F-15d.csv	1,306	4,065	2,759	53,041
USROP_A 4 N-SH_F-15Sd.csv	1,401	4,090	2,689	51,708
USROP_A 5 N-SH-F-5d.csv	2,828	3,792	964	18,548
USROP_A 6 N-SH_F-9d.csv	225	634	408	7,851

always immediately available while drilling, and it is actually missing from the Volve dataset for a number of wells. The lack of lithology information will therefore promote creation of more universal models. There is still a possibility of an ROP prediction model that identifies lithology through unsupervised methods, such as clustering first, and then applies this knowledge to the testing dataset. Such approach potentially makes the model more robust and applicable to more operations. Nevertheless, further work is planned on creating additional curated dataset that would contain lithology information, bit wear, time, and other parameters.

It is also possible to develop models that work on additional pre-defined internal splits, for example dividing data further by the wellbore diameter, so that data for a given well section is evaluated by a model created only on training data from wellbores with an identical or a similar diameter.

C.3.3 Additional information

Volve dataset contains a plethora of information about the field and all the relevant operations, ranging from daily reports and production logs to reservoir models and seismic data. Due to volume of that additional information it was not pre-processed as a part of this paper. Some of that information still may be used indirectly as a hint or idea driving model creation, or directly, where additional data is used to potentially significantly reduce prediction error. Also in this

case the USROP dataset will be useful as a reference point. When proposing an improvement, for example inclusion of seismic data to improve ROP prediction, the results may be directly benchmarked against the state-of-the-art models of other researchers. Today, if such improvement is proposed a *before-and-after* comparison is often questionable, as the *before* state is provided by the same researcher due to lack of comparable results available. This, consciously or not, may not be the best effort as one focuses on the new and improved model.

C.3.4 Score metrics

Not only data needs to be standardized in order to evaluate differences in models' performance, but unification of the way the results are calculated is also needed. Mean Absolute Error (MAE) is suggested as the key metric in case of the ROP prediction.

$$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n}|A_t - F_t| \tag{C.1}$$

Where n is the sample count, t is the consecutive sample number, A_t is actual value at sample t, and F_t is the forecast value for sample t. Rationale behind this choice is that ROP modelling is mainly done for drilling time optimization, where the interest is in the cost per meter drilled. A given value of error, for example 10 m/h, will be of roughly the same significance for an operator whether the true value is 30 m/h or 80m/h.

Commonly used alternative to MAE is Mean Absolute Percentage Error (MAPE)

$$\mathrm{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \tag{C.2}$$

This heavily penalizes errors for low ROP values. In case of an error of 10m/h in a very slow section of a well, like 0.1 m/h, becomes a 10,000% of error. In practice ROP can be very close to zero, or zero, on some datapoints, making the

problem even more extreme generating error values at infinity. Manual removal of such datapoints is possible, but would make the results significantly influenced by arbitrary decisions. This was the deciding factor behind selecting MAE as the key metric. Supplemental metric is proposed to indicate the error value related to the absolute value of ROP without the infinite error problem - Weighted Mean Absolute Percentage Error (WMAPE)

$$\text{WMAPE} = \frac{\sum_{t=1}^{n} |A_t - F_t|}{\sum_{t=1}^{n} |A_t|} \tag{C.3}$$

This way of calculating error gives an indication of the scale of the error related to the complete well without the infinite error problem, as it avoids dividing by values close to zero. WMAPE is used as a supplemental metric in this paper.

Note that this is related only to model evaluation, and other metrics may be better for the purpose of training the model. Care must be taken when evaluating total MAE as it cannot be simply averaged between different wells, as the sample count is not identical. The best approach is to store absolute error values per sample from evaluating all the iterations and calculate mean at the end. To better understand distribution of ROP values histograms are provided in Figure C.2. Note that the sample counts are displayed in logarithmic scale.

C.3.5 Defined scenarios

Three scenarios are proposed to evaluate ROP prediction models **that reflect both reference well as well as continuous learning methods discussed earlier.** First, **Continuous Learning** scenario is suggested as a particularly attractive approach for real-time prediction. Each well is evaluated separately; initially first 30 meters are available for training and validation to evaluate next 30 meters. Next iteration considers first 60 meters as available and subsequent 30 meters for testing. After that 90 meters are taken and so on. Note that the last testing section will necessarily be smaller than 30 meters, as the total

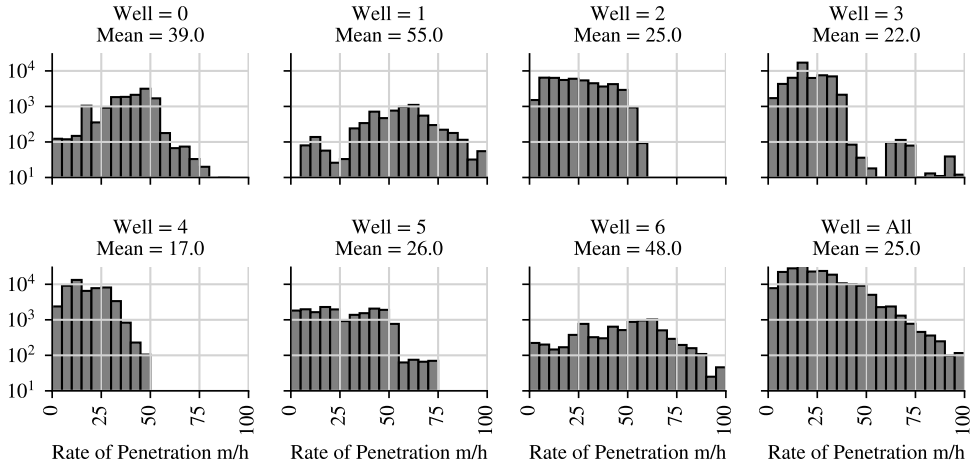


Figure C.2: ROP distribution of USROP wells.

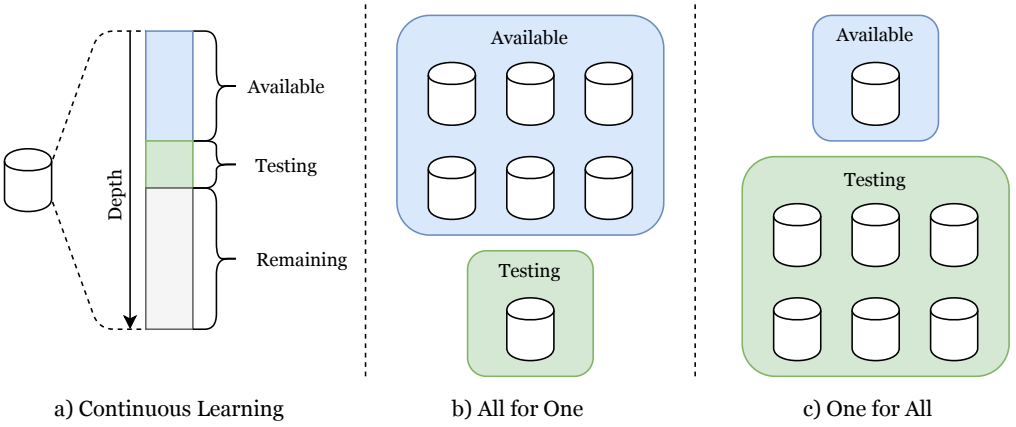


Figure C.3: Proposed scenarios for model validation

length of wells is not a multiple of 30. After processing the mean absolute error is reported for complete well taking into account testing scores from all iterations. Distance of 30m (100ft) was selected as a typical length of a stand in a common triple drilling rig. To practically implement the train/test split in a unambiguous way, the split should done be every **577**[6] depth-sorted samples, which is equivalent to approximately 30 meters of Measured Depth in USROP

[6]Total samples divided by total meters times 30; $198928/10346 \cdot 30 = 576.8 \approx 577$

dataset. This implementation works around the problem of minor gaps in the data that can potentially cause different results in different implementations.

Second proposed scenario is **All for One**[7], when all but one well is available for training and validation and one complete well is used for testing. This allows for seven different iterations with different well left for testing as cross-validation, and calls for one final MAE score from all the wells combined. Lastly, a **One for All**[8] scenario is proposed, where only one well is available, and all other wells are evaluate based on this model. As with the second scenario, this results with 7 train/test iterations acting as cross-validation. All three scenarios are shown symbolically in Figure C.3.

Note that only the split into available and testing data is fixed for each scenario. When methods like early stopping or dynamic model selection are used, the available data has to be split into training and validation data. All these samples have to be taken from the dataset designated available in the current iteration. It is also highly recommended that all publications related to ROP prediction based on this dataset share the relevant source code for research reproducibility and verification.

Note that all iterations in *All for One* and O*ne for All* scenarios are independent, hence developed algorithms should work on data only from a given iteration. In case of *Continuous Learning* scenario, each well is considered independent, but the sequence of expanding the dataset through drilling has to be maintained.

Referring to the data-split discussion in the *Incorrect data split* in the *Existing problems and potential pitfalls* section, it is worth highlighting that the proposed scenarios are *de facto* predefined, custom data-splits. Continuous Learning scenario is a variant of *TimeSeriesSplit* implementation, with fixed depth step instead of fixed split count. All for One and One for All explicitly splits the dataset into training and testing by well. Validation dataset is not specified, and it can be taken from the training data if so desired.

[7] *Training* on **All**, *testing* **for One**
[8] *Training* on **One**, *testing* **for All**

C.4 Reference results

Reference results are provided as a starting point for the basic algorithms' performance, as well as to gauge the available room for improvement. Source code to replicate the results is provided on GitHub[9]. Tested algorithms were mostly sourced from the Scikit-Learn library [28]: Gradient Boosting Regressor, Random Forest Regressor, AdaBoost Regressor and K-Nearest Neighbors Regressor. Additionally XGBRegressor was used from the popular XGBoost library [30]. Additionally, results for classical approach - Bingham model [31] developed in 1964 - are provided as well.

C.4.1 Bingham model

This is a popular model developed through laboratory testing. This is a common model acting as a reference point when suggesting improved ROP prediction methods in published research.

$$ROP = K \left[\frac{WOB}{D_b} \right]^a N \tag{C.4}$$

where K is constant accounting for formation strength, WOB is weight on bit, D_b is bit diameter, a is bit weight exponent and N is the rotary speed. The constants K and a were established based on minimizing the mean square error between the predicted and true value in the testing dataset, using the same training/testing data splits as done for the data-driven methods. A least squares optimization algorithm was used from SciPy library [32], where parameters K and a were selected such, that mean squared error (MSE), between real ROP values and calculated ROP values is smallest. This stands in apparent conflict where results are evaluated based on different metric. Our results however showed that MSE based optimization algorithms worked best producing lowest MAE and WMAPE results.

[9] https://github.com/AndrzejTunkiel/USROP

More advanced methods such as neural networks, 1D convolutions, recurrent models, automatic model selection such as TPOT library [33], ensemble results, additional pre-processing, and other approaches are possible. They are likely to yield superior results, however multiple separate publications will be needed to fully explore the ever-evolving landscape of machine learning methods that can be applied to the USROP dataset.

C.4.2 Continuous Learning

Continuous learning is an attractive practical approach that can yield good results. It does not require information from reference wells making it possible to implement it without prior knowledge or data from a given field. It also does not suffer from typical problems such as changes in logged attributes or differences in equipment used between the wells and varying equipment calibration. Reference

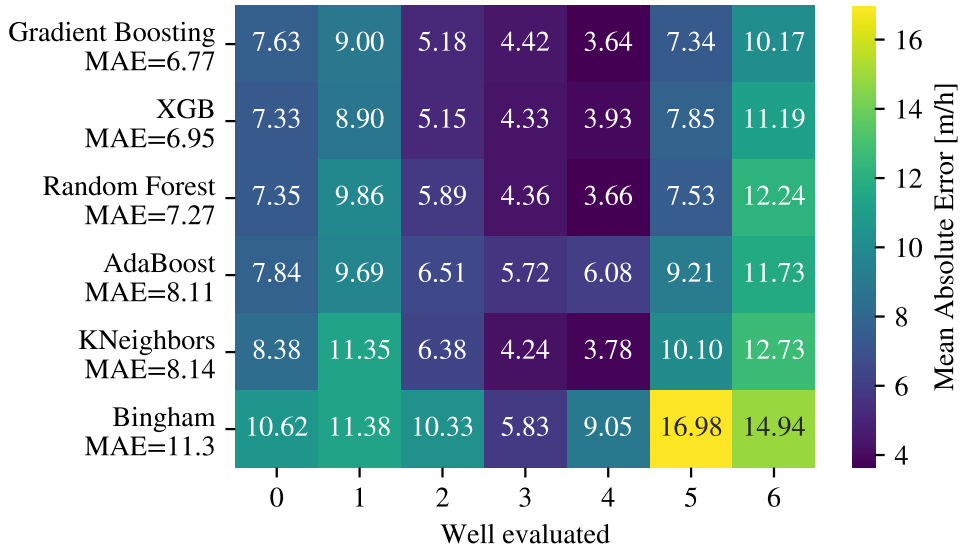


Figure C.4: Continuous learning scenario results for individual wells and methods, heatmap. Mean Absolute Error

results for this scenario are presented in Figure C.4 and Figure C.5. The main proposed metric, MAE, is shown as subfigure C.4. It is presented as a heatmap, where cells correspond to various well and method combinations, and the color represents the error value. There is a clear difference in performance between the wells, with wells 1 and 6 typically being the most poorly modelled across all applied methods, and with best results for wells 3 and 4. There are two key factors that explain this finding: wells are different lengths, with longer ones allowing for nominally bigger training datasets, and different average ROP values (ref. Table C.1 for well's size, and Figure C.2 for average ROP values). To compensate for differences in ROP, alternative heatmap is reproduced, as a subigure C.5, where results are normalized in terms of average ROP of a well and shown as WMAPE. Note that this is an error shown as a percentage of *average* ROP of a well, not ROP of a given sample, as in MAPE. This means that error of x m/h has the same value across a specific well, but it will change between wells.

For both MAE and WMAPE metrics the best overall score was achieved for

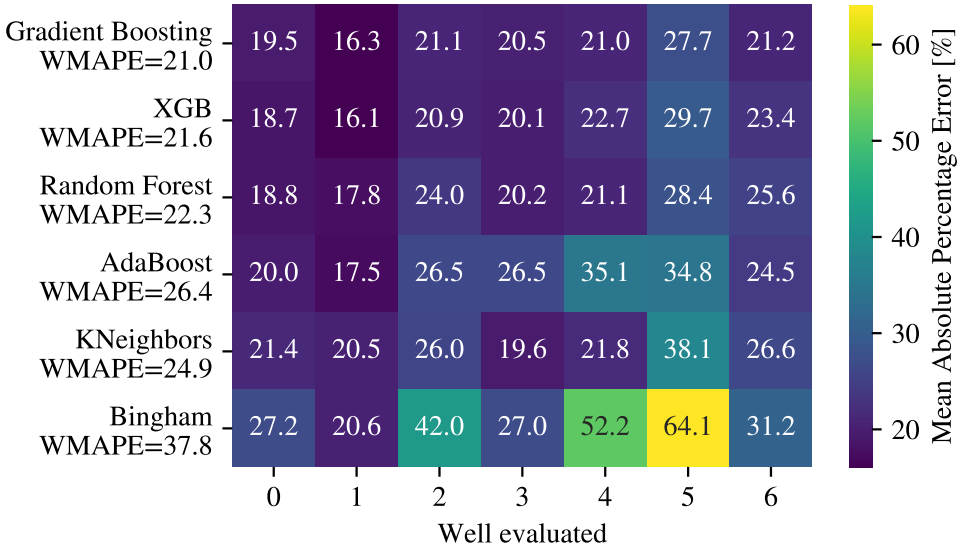


Figure C.5: Continuous learning scenario results for individual wells and methods, heatmap. Weighted Mean Absolute Percentage Error

Gradient Boosting Regressor of Scikit-Learn library. Bingham model, the classical approach, scored worst in all but one well, where it was placed next to last beating the AdaBoost Regressor, albeit only for MAE score. What is worth noting is that depending on the metric used a well can be *easy* or *difficult*. This is the case with well 1 and 6, where the error is high when calculating MAE, compared to other wells, but is low when using WMAPE. Refering back to Figure C.2, these are wells that have highest mean ROP. In practice both those metrics carry valuable information about the performance, and while there are differences between the wells, the overall rating of the methods remains mostly unchanged.

C.4.3 All for One

Reference results for All for One benchmark are shown in Figure C.6 as a violin plot. This type of chart is a merger of a histogram and a box plot, providing high resolution results at a glance in a compact format. The width of the light gray portion referencing the sample count for a given value on the y-axis, and is smoothed out via kernel density estimation. Note that the dot in the center of each figure represents median absolute error, while the key metric is mean absolute error. The dark grey bar spand 15th and 75th percentile. Best value in terms of MAE was achieved by Gradient Boosting Regressor of Scikit-Learn library. All defaults were kept for this algorithm, hence the outcomes are not fully representative of its potential. The results in terms of the well-by-well split are presented in as a heatmap in Figure C.7. It is possible to see how the tested algorithms behaved on individual wells. Note that different algorithms may perform better for different wells. While Gradient Boosting Regressor has the lowest overall error, alternative approaches worked better when testing wells 0, 1, 3, 4, and 6. This suggests that using multiple models, such as an ensemble approach, is likely to yield improvements. What is particularly noteworthy is that the classical Bingham model scored best for wells 2 and 3. Additional heatmap with WMAPE metric is shown in Figure C.8. As it was the case in Continuous Learning scenario, this metric changes for which well ROP prediction was done well or poorly, but the overall rating of tested algorithms stays the same. What

becomes highlighted however, is the fact that Bingham model's error is over 100% for well 4, which is significantly higher than other methods.

Investigating results in more detail one can notice that Continuous Learning approach typically yielded better results than All for One. While this is true for all models, it is not true for all wells. Random Forest Regressor in All for One scenario achieved better results than any model in Continuous Learning approach. While this is only a preliminary result, it suggests that different modelling approaches may be optimal for different wells.

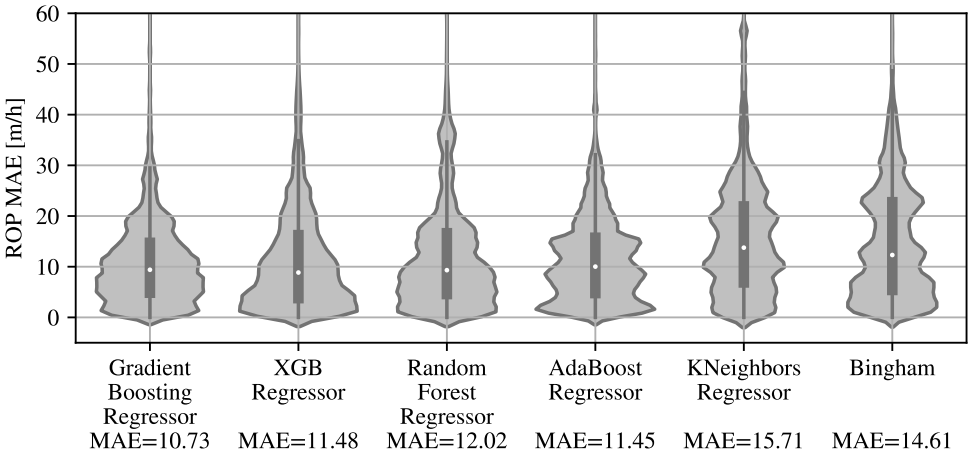


Figure C.6: All for One results per method, violin plots

C.4.4 One for All

The One for All benchmark is significantly more difficult, as it is trained on one well only in contrast to six wells in All for One variant. This is visible clearly in inferior results seen in Figure C.9. In this benchmark, again, it is the Gradient Boosting Regressor that shows the lowest MAE; albeit the value is 49% higher than in All for One scenario. The heatmap in Figure C.10 shows that, surprisingly, the best algorithm showed worst results for well 0 out of all tested algorithms.

Note that the average MAE is calculated per sample basis, hence good results in biggest wells, here 3 and 4, have the highest weight. The traditional Bingham model again showed mixed performance scoring well for some wells (5) and poorly for others (6). In calculations done for WMAPE, Figure C.11, results are similar with Bingham model performing worst, but uncovering that relatively it was the well 4 where it was most off-target, as that well was drilled more slowly, increasing the percentage error.

C.5 Discussion

C.5.1 Comparison against a flawed methodology

The reference results for all three proposed scenarios significantly differ from numbers presented in previous research related to ROP prediction using machine

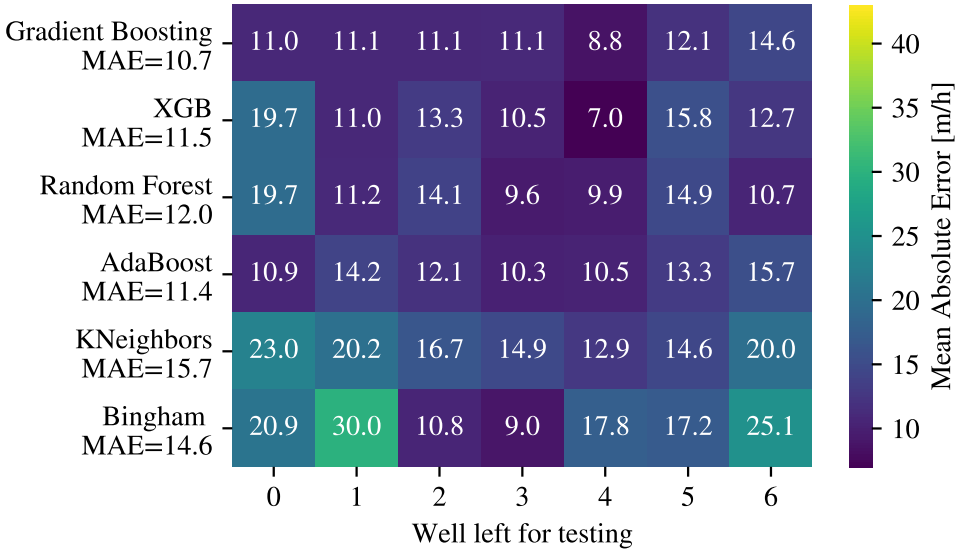


Figure C.7: All for One results for individual wells and methods, heatmap. Mean Absolute Error

learning. The absolute error is much higher, and improvements over the Bingham model are only modest. First basic reason for this situation is that presented results are not indicative of the state of the art, but act as an indication of

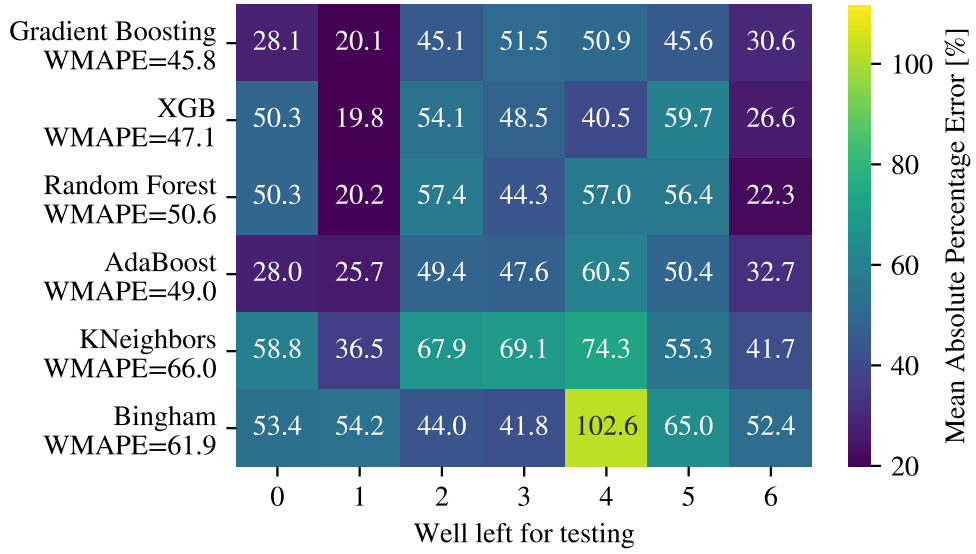


Figure C.8: All for One results for individual wells and methods, heatmap. Weighted Mean Absolute Percentage Error

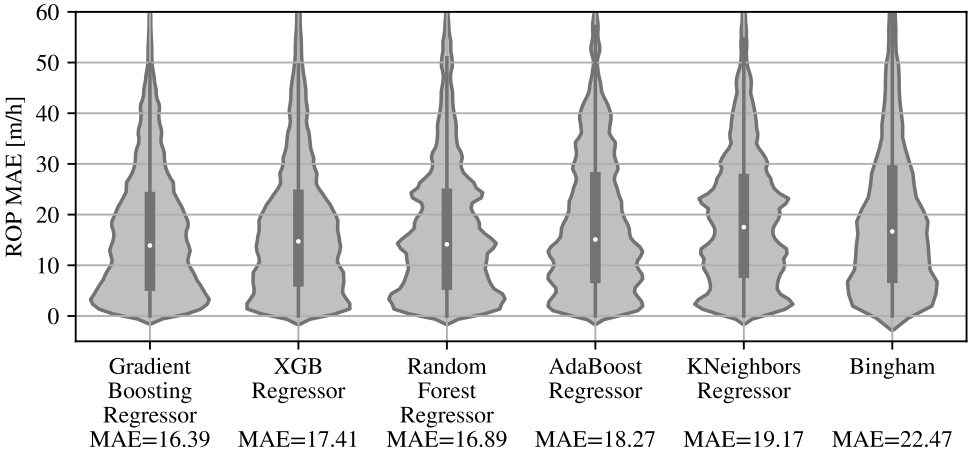


Figure C.9: One for All results per method, violin plots

performance of the off-the-shelf algorithms applied without any tuning. This paper does not aim at developing an ROP prediction model, but to facilitate comparative research in this domain.

The second reason for relatively poor results is that all three proposed scenarios were developed to be realistic and representative of the overall performance of different models. Proposed dataset consists of drilling operation through various lithologies with no explicit attribute identifying them, and using different eqiupment. This is representative of real-life drilling, where such information is not always readily available. It was found to be very common in related publications that they are often limited the ROP prediction models in specific lithology.

To underline the perceived performance differences potentially stemming from different data split an exercise was performed, where USROP dataset as a whole, all wells joined together, was split into train/test portions in *9:1* ratio, with data rows randomly assigned to those subsets. Such methodology is flawed and does

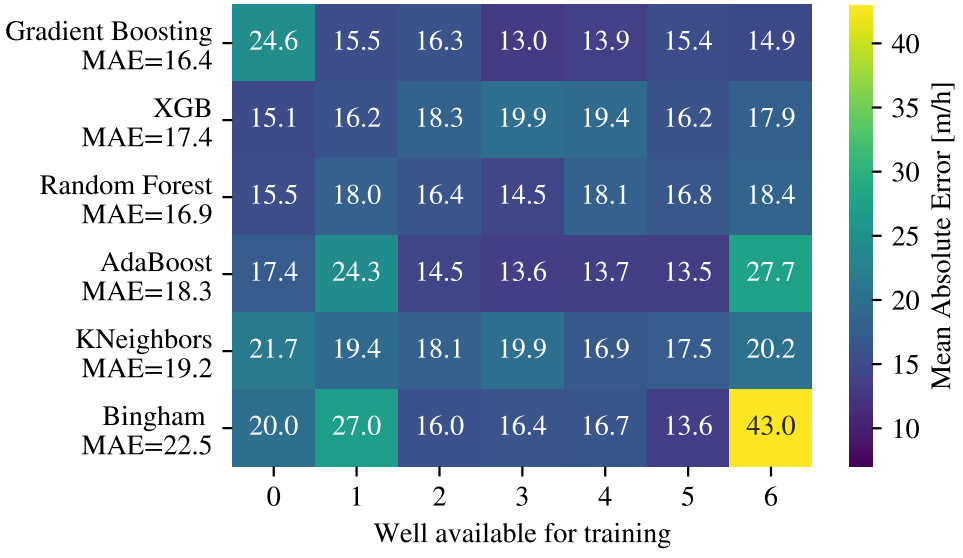


Figure C.10: One for All results for individual wells and methods, heatmap. Mean Absolute Error

not represent real-life performance, a problem indicated in the second section of this paper. This approach was found to be in use in at least one of the reviewed papers, unsurprisingly reporting very good results. Using an untuned Gradient Boosting Regressor we achieved MAE value of *4.75*, and $R^2 = 0.82$, better than all benchmarked algorithms in all the USROP proposed scenarios, and approximately half the error of the best global result. To further improve the score, TPOT library [33] was used to automatically search for best performing off-the-shelf algorithm to replicate assumed reasonable best effort in making an ROP model. This resulted in an algorithm with MAE under *0.3 m/h*. Further stretching the apparent performance by applying the train/test split with different random seeds it was possible to identify a specific random sampling pushing the MAE down to *0.265 m/h* and a near-perfect R^2 value of *0.999*. While those numbers may look impressive, they are just a result of a flawed methodology and do not translate to practical application.

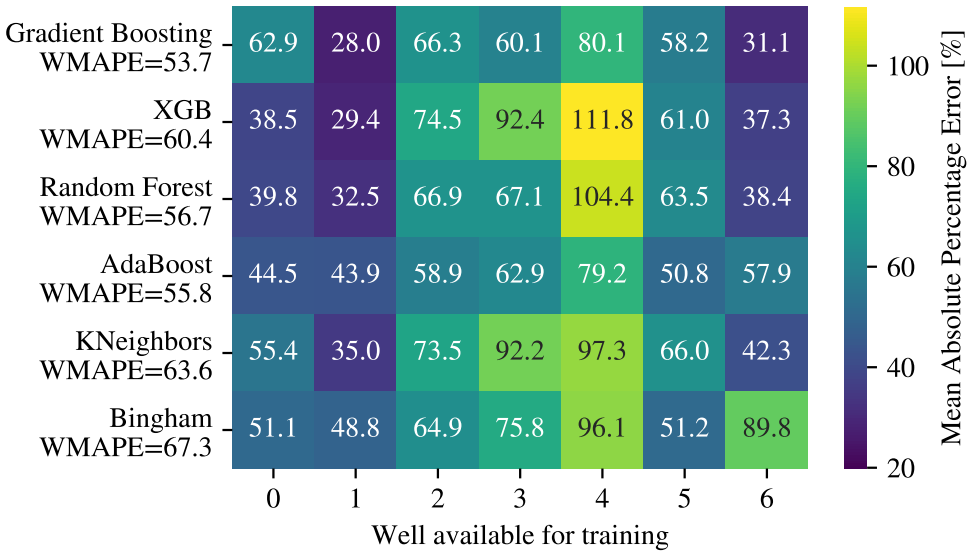


Figure C.11: One for All results for individual wells and methods, heatmap. Weighted Mean Absolute Percentage Error

C.5.2 Bingham reference results

Among the tested algorithms in this paper the Bingham model is both most widely used by other researchers, and the one implemented most unambiguously. This allows one to compare the USROP dataset to other datasets via proxy of this model. Previous work on undisclosed dataset provided by Marathon [34] evaluated the Bingham model both on individual lithologies as well as on entire dataset. The overall MAPE ranged between 23% and 43% depending on the method used to identify the coefficients. This is in line with results from the Continuous Learning scenario, where in USROP dataset the WMAPE was between 20% and 103% when inspecting individual wells in different scenatios. Another research [19] showed Bingham model achieving MAPE between 33% and 40%, again in line with results from USROP dataset. It is worth noting that these metrics are similar, yet not identical, since WMAPE in our paper refers to percentage of average ROP to avoid results being overwhelmed by moderate nominal error at very low ROPs. Methods in calculating the coefficients of the Bingham model also vary, with the quoted papers using a 3rd party software, making comparisons difficult. For the sake of comparison MAPE was also calculated per sample for All for One scenario and shown in Figure C.12. The mean error is 102%, however as expected, the histogram clearly shows that the bulk of actually expected error is between 0% and 60% with outliers inflating the average. Truncating these would significantly reduce the overall MAPE. Those results are in-line with the other quoted papers, suggesting that USROP dataset does not significantly differ in terms of ROP prediction difficulty.

C.5.3 Future work

Volve dataset, which is the source for the USROP dataset presented here, contains significant amount of additional data that was not included. This includes, but is not limited to, time-based data, pit volume, trip tank volume, block position, bit depth, mud type, mud name, mud properties, daily drilling reports, seismic data, lithology data, production data, and more. Significant effort is necessary

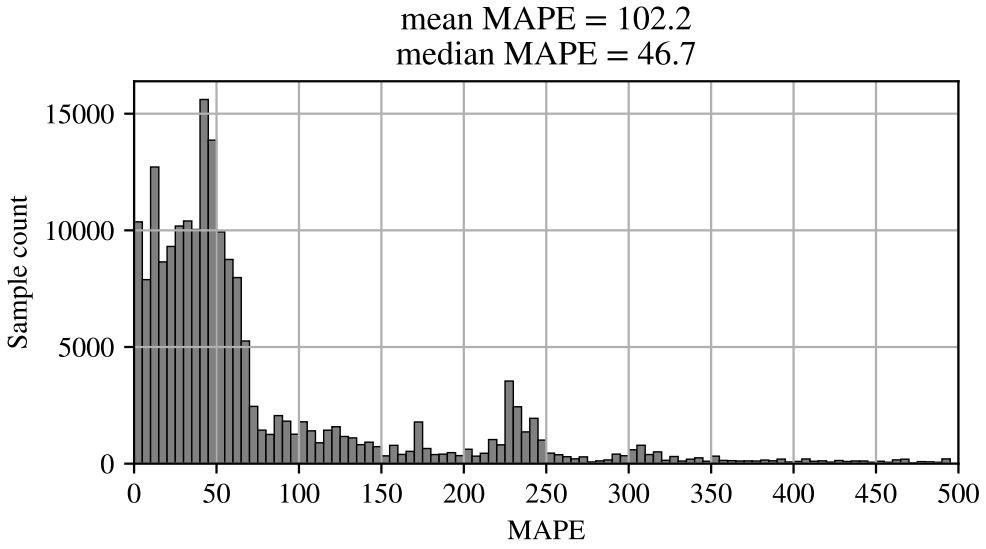


Figure C.12: Mean Absolute Percentage Error, calculated as per sample percentage (MAPE).

to make these data seamlessly available for the purpose of machine learning or more general data science. Operations in the Volve field were not recorded using unified parameter set, and therefore making a curated dataset is necessarily a balance between the amount of parameters included and the amount of wells that contain all the selected parameters.

Immediate work that is planned by authors is to either expand, or create additional dataset meant for ROP prediction that includes lithology data, as well as parameters currently missing that are necessary for implementation of Bourgoyne and Young ROP model [35]; these include jet impact force, pore pressure gradient, fractional bit tooth wear and threshold bit weight per inch of bit diameter at which the bit begins to drill. Other researchers are encouraged to create curated (sub)datasets for the specific problems they are working on, what will facilitate and accelerate research in the respective domain.

C.6 Conclusion

The key novel aspect of presented work is the creation of a reference, pre-processed, and simple to use ROP prediction dataset with specific challenges to solve has high potential to become a catalyst for higher quality research. It is a necessary step to evaluate what is the current state-of-the art in ML applied to drilling. The proposed three scenarios are related to real-life situations and aim to be representative of field deployment. As the reference dataset is based on Volve data, it is possible to extract further information about the wells used, allowing for more informed modelling decisions, background information, as well as further expansion towards more reference datasets. It allows researchers to propose inclusion of specific new information and to have a universal benchmark to show the achieved improvement.

Additionally, we found that when the data-split is performed in a realistic manner, suitable for field deployment, the standard ML algorithms perform worse than in previously published studies. Presented reference results shed a light on the performance of data-driven methods, where, depending on the methodology and specific well, they may be both vastly superior and sometimes inferior to the classical approach of physics-based models. This further confirms the need for bigger reference datasets, which enable robust evaluation of the accuracy of developed models.

```
import numpy as np
from sklearn.ensemble import GradientBoostingRegressor
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
np.random.seed(42)

X = np.linspace(0,1,10000)
y = np.cumsum(np.random.normal(size=10000))

reg = GradientBoostingRegressor(random_state=42)

X_train, X_test, y_train, y_test = train_test_split(
    X[:,np.newaxis], y, test_size=0.33, random_state=42)
```

```
reg.fit(X_train,y_train)

print(reg.score(X_test,y_test))
```

Listing C.1: Train/Test split failure

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
df = pd.read_csv('USROP_A 4 N-SH_F-15Sd.csv')
y = df['Measured Depth m'].to_numpy()
X = df[['Average Surface Torque kN.m',
        'Average Rotary Speed rpm']]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.33, random_state=42)

reg = GradientBoostingRegressor(random_state=42)
reg.fit(X_train,y_train)

print(reg.score(X_test,y_test))
```

Listing C.2: Measured Depth prediction

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
np.random.seed(42)

X = np.random.normal(20,2,size=10000)
X = X[:,np.newaxis]
y = np.random.normal(50,3,size=10000)

reg = GradientBoostingRegressor(random_state=42)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.33, random_state=42)

reg.fit(X_train,y_train)
```

```
plt.scatter(y_test,reg.predict(X_test),s=1, c="black")
print(np.average(np.abs(y_test - reg.predict(X_test))) / np.average(y_test))
```

Listing C.3: Easy target failure

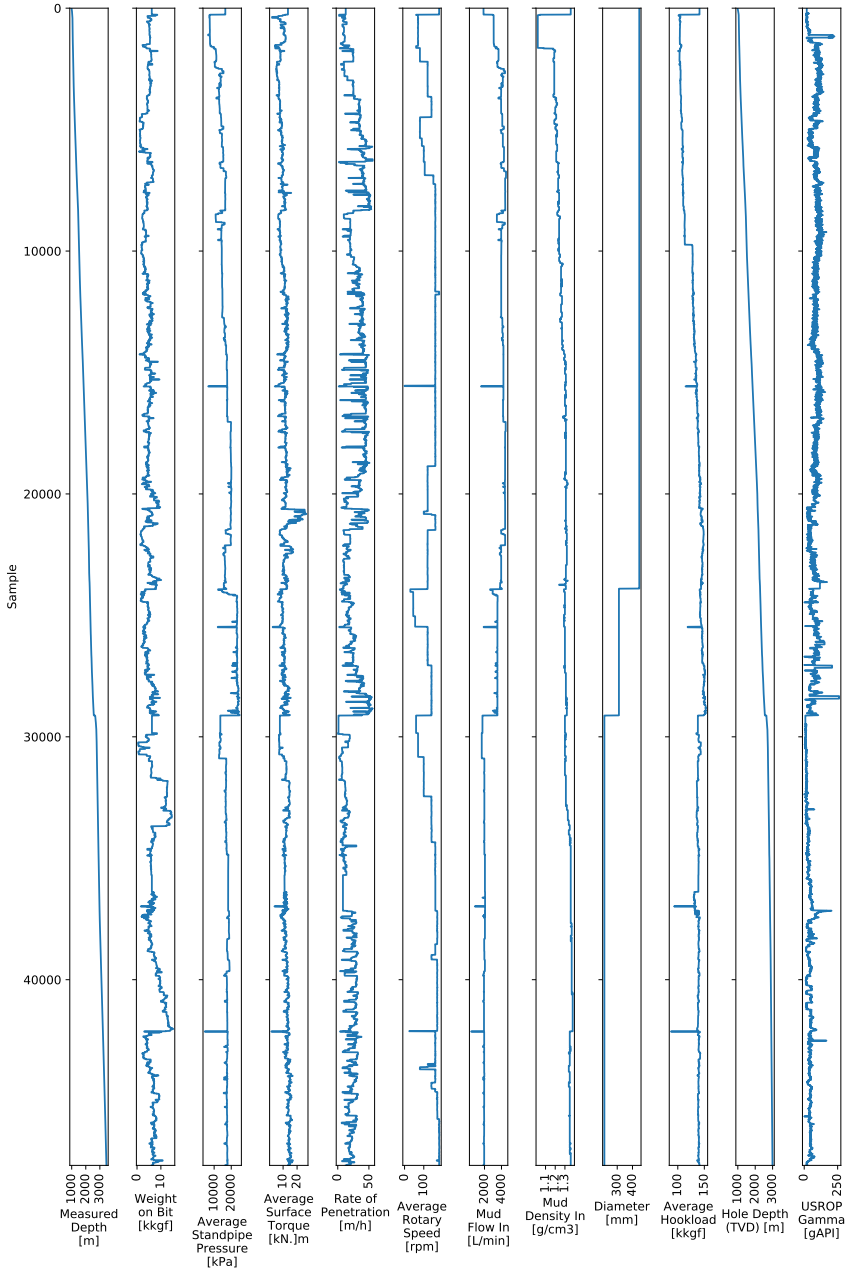


Figure C.13: All parameters of well 2, for reference only.

Bibliography

[1] George Davey Smith and Shah Ebrahim. "Data Dredging, Bias, or Confounding". In: *British Medical Journal* 325.7378 (Dec. 2002), pp. 1437–1438. ISSN: 09598146. DOI: 10.1136/bmj.325.7378.1437. pmid: 12493654.

[2] Monya Baker and Dan Penny. "Is There a Reproducibility Crisis?" In: *Nature* 533.7604 (2016), pp. 452–454. ISSN: 14764687. DOI: 10.1038/533452A.

[3] Equinor. *Volve Field Data (CC BY-NC-SA 4.0)*. 2018. URL: https://www.equinor.com/en/news/14jun2018-disclosing-volve-data.html.

[4] Andrzej T Tunkiel, Tomasz Wiktorski, and Dan Sui. "Drilling Dataset Exploration, Processing and Interpretation Using Volve Field Data". In: *ASME 2020 39th International Conference on Ocean, Offshore and Arctic Engineering* (2020). DOI: 10.1115/OMAE2020-18151.

[5] IHS Markit. *Offshore Rig Day Rate Index*. URL: https://ihsmarkit.com/products/oil-gas-drilling-rigs-offshore-day-rates.html.

[6] H. Rahimzadeh, M. Mostofi, and A. Hashemi. "A New Method for Determining Bourgoyne and Young Penetration Rate Model Constants". In: *Petroleum Science and Technology* 29.9 (Mar. 2011), pp. 886–897. ISSN: 1091-6466. DOI: 10.1080/10916460903452009. URL: http://www.tandfonline.com/doi/abs/10.1080/10916460903452009.

[7] Bing Liu. "Lifelong Machine Learning: A Paradigm for Continuous Learning". In: *Frontiers of Computer Science* 11.3 (2017), pp. 359–361. ISSN: 20952236. DOI: 10.1007/s11704-016-6903-6.

[8] Abdulmalek Ahmed, Abdulwahab Ali, Salaheldin Elkatatny, and Abdulazeez Abdulraheem. "New Artificial Neural Networks Model for Predicting Rate of Penetration in Deep Shale Formation". In: *Sustainability (Switzerland)* 11.22 (Nov. 2019), p. 6527. ISSN: 20711050. DOI: 10.3390/su11226527.

[9] Chiranth Hegde and K. E. Gray. "Use of Machine Learning and Data Analytics to Increase Drilling Efficiency for Nearby Wells". In: *Journal of Natural Gas Science and Engineering* 40 (2017), pp. 327–335. ISSN: 18755100. DOI: 10.1016/j.jngse.2017.02.019.

[10] Chiranth Hegde, Scott Wallace, and Ken Gray. "Using Trees, Bagging, and Random Forests to Predict Rate of Penetration during Drilling". In: *Society of Petroleum Engineers - SPE Middle East Intelligent Oil and Gas Conference and Exhibition* (2015). DOI: 10.2118/176792-ms.

[11] Chiranth Hegde, Hugh Daigle, Harry Millwater, and Ken Gray. "Analysis of Rate of Penetration (ROP) Prediction in Drilling Using Physics-Based and Data-Driven Models". In: *Journal of Petroleum Science and Engineering* 159 (Nov. 2017), pp. 295–306. ISSN: 09204105. DOI: 10.1016/j.petrol.2017.09.020.

[12] Cesar Soares and Kenneth Gray. "Real-Time Predictive Capabilities of Analytical and Machine Learning Rate of Penetration (ROP) Models". In: *Journal of Petroleum Science and Engineering* 172 (Jan. 2019), pp. 934–959. ISSN: 09204105. DOI: 10.1016/j.petrol.2018.08.083.

[13] Chiranth Hegde and Ken Gray. "Evaluation of Coupled Machine Learning Models for Drilling Optimization". In: *Journal of Natural Gas Science and Engineering* 56 (Aug. 2018), pp. 397–407. ISSN: 18755100. DOI: 10.1016/j.jngse.2018.06.006.

[14] Mohammad Sabah, Mohsen Talebkeikhah, David A. Wood, Rasool Khosravanian, Mohammad Anemangely, and Alireza Younesi. "A Machine Learning Approach to Predict Drilling Rate Using Petrophysical and Mud Logging Data". In: *Earth Science Informatics* 12.3 (Sept. 2019), pp. 319–339. ISSN: 18650481. DOI: 10.1007/s12145-019-00381-4.

[15] Jiahang Han, Yanji Sun, and Shaoning Zhang. "A Data Driven Approach of ROP Prediction and Drilling Performance Estimation". In: *International Petroleum Technology Conference 2019, IPTC 2019* (2019). DOI: 10.2523/iptc-19430-ms.

[16] Xian Shi, Gang Liu, Xiaoling Gong, Jialin Zhang, Jian Wang, and Hongning Zhang. "An Efficient Approach for Real-Time Prediction of Rate of Penetration in Offshore Drilling". In: *Mathematical Problems in Engineering* 2016 (2016). Ed. by Cheng-Tang Wu, p. 3575380. ISSN: 1024-123X. DOI: 10.1155/2016/3575380.

[17] B. Mantha and R. Samuel. "ROP Optimization Using Artificial Intelligence Techniques with Statistical Regression Coupling". In: *Proceedings - SPE Annual Technical Conference and Exhibition*. Vol. 2016-Janua. Society of Petroleum Engineers (SPE), Sept. 2016. ISBN: 978-1-61399-463-4. DOI: 10.2118/181382-ms.

[18] Tuna Eren and Mehmet Evren Ozbayoglu. "Real Time Optimization of Drilling Parameters during Drilling Operations". In: *SPE Oil and Gas India Conference and Exhibition*. Society of Petroleum Engineers, Apr. 2010. DOI: 10.2118/129126-MS.

[19] Cesar Soares, Hugh Daigle, and Ken Gray. "Evaluation of PDC Bit ROP Models and the Effect of Rock Strength on Model Coefficients". In: *Journal of Natural Gas Science and Engineering* 34 (Aug. 2016), pp. 1225–1236. ISSN: 18755100. DOI: 10.1016/j.jngse.2016.08.012.

[20] Omogbolahan S. Ahmed, Ahmed A. Adeniran, and Ariffin Samsuri. "Computational Intelligence Based Prediction of Drilling Rate of Penetration: A Comparative Study". In: *Journal of Petroleum Science and*

Engineering 172 (Jan. 2019), pp. 1–12. ISSN: 09204105. DOI: 10.1016/j.petrol.2018.09.027.

[21] Khoukhi Amar and Alarfaj Ibrahim. "Rate of Penetration Prediction and Optimization Using Advances in Artificial Neural Networks, a Comparative Study". In: *IJCCI 2012 - Proceedings of the 4th International Joint Conference on Computational Intelligence*. 2012, pp. 647–652. ISBN: 978-989-8565-33-4. DOI: 10.5220/0004172506470652.

[22] Tuna Eren and Evren Ozbayoglu. "Real-Time Drilling Rate of Penetration Performance Monitoring". In: *Offshore Mediterranean Conference and Exhibition, 23-25 March, Ravenna, Italy*. Offshore Mediterranean Conference, 2011. DOI: OMC-2011-076.

[23] A. T. Bourgoyne and F. S. Young. "A Multiple Regression Approach to Optimal Drilling and Abnormal Pressure Detection". In: *SPE Reprint Series* 14.49 (Aug. 1999), pp. 27–36. ISSN: 08910901. DOI: 10.2118/4238-pa.

[24] Ping Yi, Aniket Kumar, and Robello Samuel. "Real-Time Rate of Penetration Optimization Using the Shuffled Frog Leaping Algorithm (SFLA)". In: *Society of Petroleum Engineers - SPE Intelligent Energy International 2014*. Society of Petroleum Engineers (SPE), Apr. 2014, pp. 116–125. ISBN: 978-1-63266-413-6. DOI: 10.2118/167824-ms.

[25] Wanyi Jiang and Robello Samuel. "Optimization of Rate of Penetration in a Convoluted Drilling Framework Using Ant Colony Optimization". In: *SPE/IADC Drilling Conference, Proceedings*. Vol. 2016-Janua. Society of Petroleum Engineers (SPE), Mar. 2016. ISBN: 978-1-61399-401-6. DOI: 10.2118/178847-ms.

[26] Li Deng. "The Mnist Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[27] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. "EMNIST: An Extension of MNIST to Handwritten Letters". In: *Arxiv preprint* (Feb. 2017). URL: http://arxiv.org/abs/1702.05373.

[28] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. "Scikit-Learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[29] Chiranth Hegde, Harry Millwater, Michael Pyrcz, Hugh Daigle, and Ken Gray. "Rate of Penetration (ROP) Optimization in Drilling with Vibration Control". In: *Journal of Natural Gas Science and Engineering* 67 (July 2019), pp. 71–81. ISSN: 18755100. DOI: 10.1016/j.jngse.2019.04.017.

[30] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Vol. 13-17-Augu. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785.

[31] M.G. Bingham. *A New Approach to Interpreting Rock Drillability*. The Petroleum Publishing Co., 1964.

[32] Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[33] Randal S. Olson, Ryan J. Urbanowicz, Peter C. Andrews, Nicole A. Lavender, La Creis Kidd, and Jason H. Moore. "Automating Biomedical Data Science through Tree-Based Pipeline Optimization". In: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9597. Springer Verlag, 2016, pp. 123–137. ISBN: 978-3-319-31203-3. DOI: 10.1007/978-3-319-31204-0\{_\}9.

[34] Cesar Mattos de Salles Soares. "Development and Applications of a New System to Analyze Field Data and Compare Rate of Penetration (ROP) Models". University of Texas at Austin, 2015.

[35] A T Bourgoyne Jr, K K Millheim, M E Chenevert, and F S Young Jr. "Applied Drilling Engineering. Volume 2". In: (1986).

Appendix D

Continuous Drilling Sensor Data Reconstruction and Prediction via Recurrent Neural Networks, AT3

Tunkiel, A. T., Wiktorski, T., & Sui, D. (2020). **Continuous drilling sensor data reconstruction and prediction via recurrent neural networks**. In ASME 2020 39th International Conference on Ocean, Offshore and Arctic Engineering. American Society of Mechanical Engineers Digital Collection.

Appendix E

Training-while-drilling approach to inclination prediction in directional drilling utilizing recurrent neural networks, AT4

Tunkiel, A. T., Sui, D., & Wiktorski, T. (2021). **Training-while-drilling approach to inclination prediction in directional drilling utilizing recurrent neural networks**. Journal of Petroleum Science and Engineering, 196, 108128.

Abstract

Machine Learning adoption within drilling is often impaired by the necessity to train the model on data collected from wells analogous in lithology and equipment used to the well where the model is meant to be deployed. Lithology information

is not always well documented and fast-paced development of drilling equipment complicates the challenge even further, as a model would likely become obsolete and inaccurate when new technologies are deployed. To bypass this problem a training-while-drilling method utilizing neural networks that are capable of modelling dynamic behaviour is proposed. It is a continuous learning approach where a data-driven model is developed while the well is being drilled, on data that is received as a continuous stream of information coming from various sensors. The novelty in presented approach is the use of Recurrent Neural Network elements to capture the dynamic behaviour present in data. Such model takes into account not only values of the adjacent data, but also patterns existing in the data series. Moreover, results are presented with a focus on the continuous learning aspect of the method, which was sparsely researched to date. A case study is presented where inclination data is predicted ahead of the inclination sensor in a directional drilling scenario. Our model architecture starts to provide accurate results after only 180 meters of training data. Method, architecture, results, and benchmarking against classical approach are discussed; full dataset with complete source code is shared on GitHub.

E.1 Introduction

Lack of adequate training data is one of the major issues preventing machine learning model deployment within petroleum. While in general it is relatively easy to develop data-driven models for problems like rate of penetration (ROP) prediction, such models will be valid only for wells where geology, equipment, and general design matches closely the training dataset. This is further corroborated by the lack of published general-purpose data-driven ROP prediction models. All machine learning models face such challenge; if an algorithm is trained to detect cats, but the dataset contains only cats indoors, it will struggle to classify pictures taken outdoors. Such problem was explored in practice when a neural network was trained to discern dogs from wolves. Training dataset was made flawed on purpose, where pictures of dogs were taken on grass, and pictures of

wolves in the snow. This lead to the classifier using snow as the key feature, and subsequently poor model performance [1].

To solve this underlying issue, continuous learning [2] methods could be used, where a model is continuously retrained while the well is being drilled. Data collected from a drilled section are used to train a model that can be applied to the further section of the same well. When additional section of a well is drilled, the process is repeated to create an updated model. Advances in computational power make data-driven model training time negligible in comparison to time required to drill a well making the training-while-drilling approach feasible. Model training is often fast enough to be completed in the short breaks in the drilling process, such as adding a stand to the drillstring. Additional benefit of a dynamically trained model is that any discrepancies between predictions and incoming data are used as a feedback to improve the subsequent iteration of the model.

There is limited previous research related to such approach in drilling. Data-driven rate of penetration (ROP) prediction models are abundant in the latest literature: [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 12, 16, 17]. There is however limited work related to continuously expanding dataset. Only few papers were identified where the training to testing ratio was explored showing improvement over analytical methods even at smallest training datasets [18]. Application of continuous expanding of the training dataset was researched as well in other papers, such as [4], applying random forest algorithm to again predict the ROP, and [19], where expanding dataset was used for ROP prediction implemented as changing train/test ratio, evaluating random forest, support vector machines and neural networks, and comparing it to analytical models such as Bingham, and Bourgoyne and Young. No other analysis of continuous learning in drilling environment was identified.

To expand on this existing work, a novel model was developed that uses not only the real-time attributes as inputs from a specific time and space, but also utilizes previous values; this is what this paper refers to as dynamic behaviour. It means that the model is aware of not only the current state, but also of the previous values and how they change along the data series, be it space or time,

identifying the dynamics of the local environment. This is achieved through the use of Recurrent Neural Network (RNN) [20], where attribute values are fed to the network from multiple steps along the data series.

To the best of our knowledge, no drilling related continuous learning research was done that utilized the recurrent neural networks the way this paper proposes. This paper also performs a thorough analysis of how the models' performance change as the data is continuously acquired; we were unable to identify any drilling-related paper that would discuss this aspect in a comparable detail.

While our novel approach does not produce results from the first meter drilled, it requires relatively small dataset to start working reliably. A case study is presented where lagging inclination data is predicted in a directional drilling scenario using a bent sub. It was selected because the problem is sparsely explored in the existing research, and the way that data behaves makes it a good candidate for a neural network model with recurrent elements. The applied model is based on our earlier work [21] where the problem of predicting lagging inclination data was first explored. In this paper, accuracy along the depth of the well is explored to evaluate method's usefulness and applicability in real-life situations.

E.1.1 Motivation

In the recent years, directional drilling became one the common drilling methods, especially in relation to shale developments[22]. Precise well placement is an important factor when it comes to the future well performance. Directional driller depends on the values from downhole sensors to know where the well is being placed. One of the challenges is, that due to space constraints, those sensors are at a significant distance from the bit, often tens of meters. This in turn creates a blind zone, a section of a well that is drilled, but the driller does not know where it exactly is, potentially leading to a delayed corrective actions.

As the sensor data is delayed, decisions taken based on these sensors' readings are delayed as well, leading to suboptimal well placement. With pay zones only 5

- 15m thick, as in case of the Bakken field[23], minimizing that delay distance in the directional readings is critical. The goal of this case study is to predict such continuous inclination readings that are yet to be made, predicting the well direction between the sensor and the bit.

E.1.2 Innovation

There are a number of innovative elements in the presented paper. Only one prior published study was identified discussing the recreation of sensor data using machine learning methods, apart from parts of the proposed method presented by the authors on the *39th International Conference on Ocean, Offshore & Arctic Engineering* in August 2020 [21]. Presentation was given [24] on similar topic applying basic regression algorithms, lasso, ridge, random forest and gradient boosting, to predict a number of sensor values lagging behind the bit. Achieved results showed relative error less than 16% for 80% of the tested data. Our research uses more advanced network architecture as well as is considered within continuous learning environment. Applying machine learning allows for method deployment when prior specific knowledge of the bit steering mechanism is not necessary. Such exact information is on the other hand needed to follow recently published analytical approach, such as performed by [25, 26], where beam bending model is developed based on exact bottom hole assembly geometry and function.

Another key innovative element presented in the case study is the application of *continuous learning*. This concept is related to lifelong machine learning [2], where continuously expanding training dataset is used to evaluate samples from the immediate future. While there is significant research related to data stratification, i.e. the split ratio between training and testing datasets, such as [27, 28], it must be noted that this is a similar, yet different topic. Continuous learning mimics the real life learning, where immediate future is predicted using all the past experiences, while stratification studies consider a fixed dataset and the best way to split it. Presented case study focuses of the models' performance in

the continuous learning scenario in detail, which we were unable to identify in literature.

Lastly, inclusion of past values as inputs via use of recurrent neural networks is also a topic sparsly explored in research related to drilling. Publications related to flow rate estimation [29] utilized generic recurrent neural networks, as well as newer work on kick detection [30] utilized newer architecture of Long-Short Term Memory. Our work expands on this by utilizing Gated Recurrent Units, RNN cell first discussed in 2014 [31] in a continuous learning scenario, a combination that we were unable to identifiy in literature related to drilling.

The proposed solution is fast to deploy, requires no proprietary software and can be run using any modern consumer-grade Graphics Processing Unit (GPU), making the necessary investment very low. Properly set up system automatically adapts to available data through dimensionality reduction techniques discussed in the further chapters. A single well data is required to validate the method for a given use case. Given the auxiliary nature of the generated results, there is little to no risk in deploying the presented method to the field. The accuracy of the method can be continuously monitored, since true values are measured with 23 meter lag relative to the prediction.

E.1.3 Machine Learning methods used

Machine learning can be applied in various ways. Generally speaking, an algorithm learns the correlations between inputs and outputs that can later be exploited for prediction purposes. One of the methods of implementing this is to use data from a given moment in time to predict a different, unknown parameter. For example, weight on bit (WOB) and drill bit's rpm can be correlated with rate of penetration, so that optimization can be done on the developed model to maximize the ROP. That correlation can be captured using various algorithms, such as linear regression, decision trees, neural networks, gradient boosting and others. This approach will however not capture any dynamic behaviour of the model. This can be rectified partially by calculating derivatives of inputs, but it will have a very limited impact. To fully capture dynamic behaviour of a given

model Recurrent Neural Networks[20] are used. This is an architecture suited for data-series, such as speech, language processing, or drilling logs. Its internal structure is well suited to take inputs both from the current state as well as a number of previous states. It contains a connection that feeds the output from step *t-1* to step *t*. The basic principle is shown in Figure E.1 on the left hand side. Practically this type of network is implemented in an unfolded form, seen on the right hand side. Input x_0 generates output h_0. At the next step, the network is fed both input x_1 as well as the output h_0, generating new output h_1. The actual model of the case study uses Gated Recurrent Units [32] as its RNN component. This architecture was found to perform well on relatively small datasets [31], which is a key requirement for the training while drilling approach, where dataset gradually grows from empty while the well is drilled.

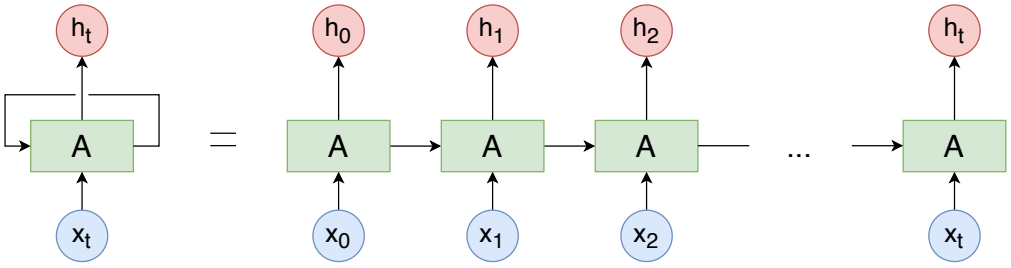


Figure E.1: Basic Recurrent Neural Network schematic

Another important aspect of developing machine learning model is how the training and testing datasets are created. This is especially important in work related to drilling, where logs are data-series. Most common way of creating a train/test split is random sampling, where a percentage of a dataset is randomly selected to be a part of a training or testing. This is a method that cannot be used for predictive models in drilling, since spurious correlations will inflate the testing result. Correct approach is to split the data into continuous sections, where first $n\%$ of a well is used as training, and remainder is used as testing. This is the most common way of performing a data split in research related to drilling.

A relatively new approach is continuous learning [2], where training dataset is continuously growing, and predictions are done based on training on all previous data. This approach fits field deployment particularly well, because it is equivalent

to how data is collected while drilling. In this approach initial results are poor due to small size of the dataset, but the assumption is, that while the dataset expands the model will outperform models created on data from offset wells, as it better represents drilling currently at hand. Figure E.2 is meant to visually explain the data split strategies discussed above.

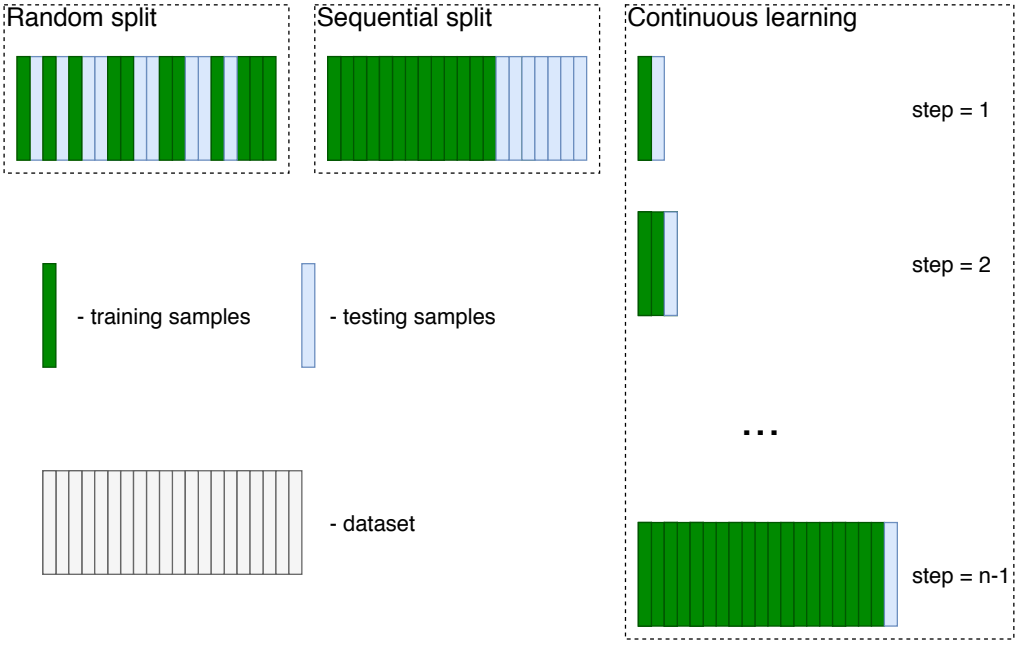


Figure E.2: Train/test split strategies

E.2 Case study and model design

The case study data from the open Volve dataset [33], [34] was used, specifically the well F9A. It was chosen as it contained a relatively long section of the well without any data issues in its depth-based log. It contains a curved section drilled with a bent sub motor, where inclination rises and falls in waves, as is characteristic of this method, see Figure E.3 for reference. The sensor lag is introduced artificially in the data and is equal to 23 meters, a value that is in range of a typical BHA configration. This was necessary as the log in question contained already depth-corrected data, an operation that is performed after the

well is drilled, hence a reverse operation was needed for a case study. What the model predicts is the continuous inclination data between the sensor and the bit location of each sample. Real-time attributes are the input to the model, including Rate of Penetration, and Weight on Bit from all the locations behind the bit, hence overlapping with the continuous inclination prediction. Inclination from the locations behind the sensor is used as in input to the RNN portion of the network. This is explained in detail in further sections.

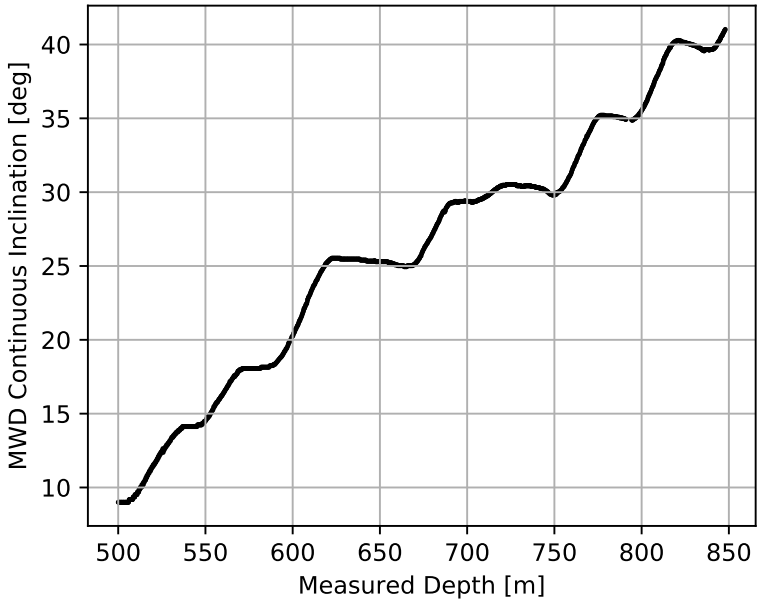


Figure E.3: Case study well inclination profile

E.2.1 Data preparation

Raw data from the real-time drilling logs are rarely usable as-is. A number of processes were applied to increase their quality. First, a section of the well without any missing data was identified, in this case, between 500 and 848 meters measured depth. Since our approach relies on neighbouring data in the model as an input, depth-steps in the data series had to be made even. RadiusNeighbourRegressorm

part of scikit-learn [35] was used to re-sample the data at even depth intervals of 0.230876 meters - median distance between datapoints in the original dataset. Attributes that have missing data after resampling process are considered not complete enough and disregarded. If a section of the well is missing some attributes, it will get discarded from future predictions. Alternatively, one can develop a system where such section of the well may be ignored completely in the process to retain certain attributes in the model when they come back on-line.

To include the past values information a windowing process was applied. Referring to Figure E.4, a single input sample contains inclination data from behind the sensor (already measured inclination values), as well as real-time attributes from behind and ahead of the sensor. In the presented case study, the distance between the sensor and the bit is 23 meters, divided into 100 discrete measurements. Distance behind the sensor taken as an input the model is also 23 meters, divided into 100 discrete measurements. The output of the model is inclination values between the sensor and the bit, also 23 meters and 100 discrete values. Referring back to Figure E.4, distances p and b are equal to 23 meters; number of discrete steps, both n and m is equal to 100. The distance between steps is even and approximately 0.23 meters. This setup creates a model with a high number of inputs and outputs. Each included real-time attribute adds 200 inputs, since there are 100 values before and 100 values after the sensor. There are also 100 inputs related to inclination values. Presented case study has 51 usable real-time attributes. These are however reduced to 3 attributes through principle component analysis (PCA), described in further subsection, resulting in practice in $3x(100 + 100) + 100 = 700$ inputs to the machine learning algorithm itself.

PCA transformation In relation to input attributes, to simplify selection process, and easy field deployment, all instantaneously available attributes are used. Inclination data is stored separately, while all other data is compressed using Principle Component Analysis [36], a dimensionality reduction method. Note that this reduces dimensions that the machine learning algorithm is exposed to only, as the input to the complete setup still takes all attributes. Resampled data is first normalized to a range (0,1), fed through a PCA algorithm that

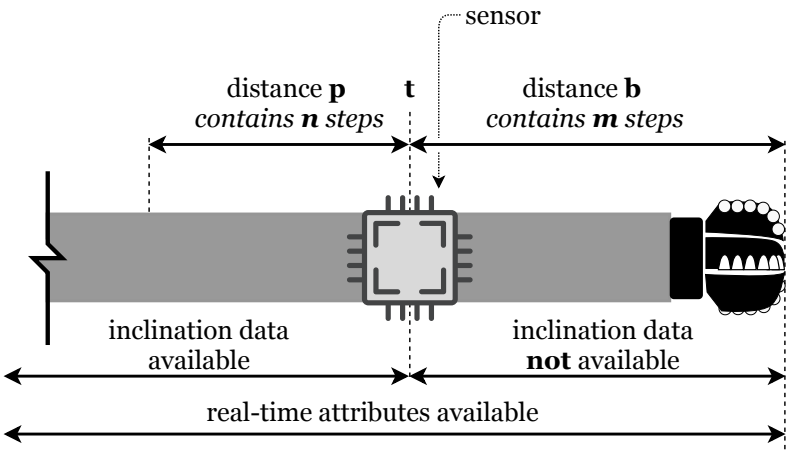


Figure E.4: Sensor lag

reduces it to the prescribed amount of components, and normalized again to a range of (0,1). The number of output attributes and how it affects the prediction was evaluated and it was found that a reduction to 3 components from initial 51 attributes[1] generates best results (Mean Square Error (MSE)=0.035) in terms of prediction error. The study for determining optimal number of PCA components was performed through complete training-while-drilling exercise, from 15% to 80% of available data, with 1% increments - process explained in detail further in the paper. Results were on average better than selecting all the attributes without PCA dimensionality reduction (MSE=0.041). PCA-based results were also better than manual selection of attributes based on engineering judgement - approach applied to a related case study before [21] (MSE = 0.048), where average surface torque, average rotary speed, and rate of penetration were selected as inputs. Data from PCA dimension evaluation results are shown in Figure E.5, where mean square training error is plotted against the number of PCA components used, plus the reference values. The best solution, with 3 components, explains 88% of the total variance. It is worth noting that standardization of data was not performed. This process of substracting mean from sample values was tried

[1]These are attributes such as Weight on Bit kkgf, Average Standpipe Pressure kPa, Average Surface Torque kN.m, Rate of Penetration m/h, etc.,

through using RobustScaler, a solution from the Sklearn package [35], and it produced overall inferior results.

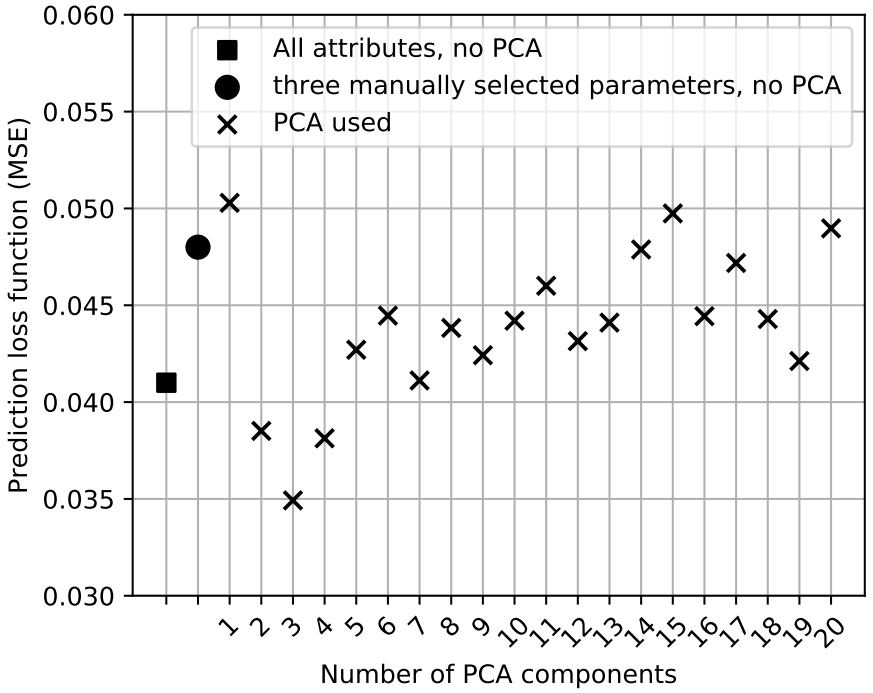


Figure E.5: PCA dimension evaluation.

The reason why dimensionality reduction decreases the error of a model is most likely tied to overfitting and spurious correlations. As explained earlier, inclusion of each real-time attribute in our case study increases number of inputs by 200. This results in 10 000 inputs if 50 attributes were to be used. Such high number of inputs in a dataset as (relatively) small as ours is bound to cause overfitting to some extent.

It must be noted that no prior attribute selection was performed. No correlation matrices were calculated nor any other approach was applied. This is connected to the expected deployment of the method, were decision related to which attributes will be available during drilling operation is not always known much in advance. Attribute selection is not trivial, and methods, such as mentioned correlation analysis are difficult to implement to work automatically; furthermore, the basic

correlation methods will uncover only linear relationships. Therefore using all the available parameters through the PCA transformation is proposed as a solution that can be done fully automatically without manual intervention.

Nominal and incremental inclination data Preparation of inclination data was different than for other parameters. It is not immediately obvious if best results will be achieved while predicting inclination data itself, or change in inclination (incremental value ,first derivative), therefore both approaches were evaluated in parallel. Use of inclination change is simpler, as it can be used directly with (0,1) normalization. Use of actual inclination data is more difficult, as it requires normalization through introduction of a local coordinate system. Proposed neural network uses RNN layer to process previous values of the predicted attribute; in our case *n=100* input steps were selected, a value selected through hyperparameter tuning, which is explained in detail later in this paper. Nominal inclination data have to be scaled such, that first and oldest inclination input value is zero in the local coordinate system, and the highest value is no bigger than one. The length of the dataset after complete preparation is 1486 samples, a value that is a function of well depth data at hand, resampling rate, and the n and m values of the model described in the previous section.

E.2.2 Model design

Overall architecture The model consists of two branches, RNN branch and Multi-Layer Perceptron (MLP) [37] branch; these branches respectively contain additional Gaussian noise and dropout layers. They are later concatenated and connected into a single Dense layer. See Figure E.6 for reference, as well as the publication first discussing this general model [21]. All the layers used are from Keras library [38] and therefore specific details can be found in the project's documentation. Inclination data is fed into the RNN branch, and all real-time attributes are fed into the MLP branch. The model was implemented

in TensorFlow 2.1.0 with Keras library. Full source code used for this paper's case study is available on Github[2].

As indicated in the data preparation section of this paper, it is not immediately clear if in the presented case study one should predict nominal inclination or the change in inclination, later referred as *incremental* method, as opposed to *nominal* method. Neither of the methods predict the actual inclination, since individual samples are encapsulated in a local coordinate system and scaled in range (0,1). The difference is that the incremental method works on the first derivative along the depth of the inclination. In practice, the *nominal* method predicts the value of the inclination in the local coordinate system, while the *incremental* method predicts the change in inclination value in the same local coordinate system. There are pro's and con's to each of the methods, which are highlighted in the results section of this paper.

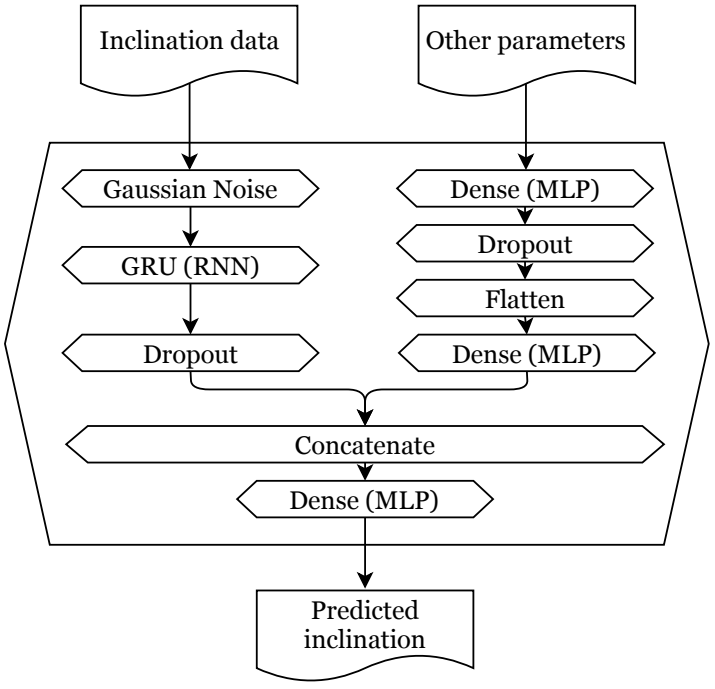


Figure E.6: Neural network architecture

[2]https://github.com/AndrzejTunkiel

E.2.3 Continuous learning implementation

Continuous learning implementation flowchart is displayed in Figure E.7. Evaluation process starts at 15 percent of available dataset, or 52 meters of drilling data. This minimum value was selected as the data from the drilled section has to be split further into training and validation. 15 percent of our dataset contains only 195 samples which are further split into *195*80%* samples for training and *195*20%* for validation. This is already a small dataset and it was decided not to train data on even smaller sample size, hence starting point of 15% was selected. Training and validation subsets are continuous and the validation data boarders with the recent end of the data; alternative strategies were tested for locating the validation data, and the best results were achieved when it was placed at the end. This split is necessary to implement early stopping, another method crucial for avoiding overfitting. The validation data are not used in the backpropagation part of the training process itself, but they are continuously evaluated while the model is trained. Typically validation error drops together with training error along the training epochs, but at the point where overfitting begins, it starts to increase. This is the point where training is stopped and the model with best validation score is retained. Data consisting of future 20 percent of the dataset is set aside for testing of the model from current iteration. 20% is relatively big, and it was chosen to be indicative of a wider model performance. It is also important to mention that the PCA dimensionality reduction model is fit only on the available data within an iteration, and not on the testing data, as it is considered not available at the time of training. In other words, the PCA transformation rules (calculating the data covariance) are established only on the part of the dataset that is considered known. Subsequent transformation is done on the dataset that contains the testing data. The inclination values are not a part of PCA transformation. The PCA model is later used for model evaluation, as the input data have to be processed with the same PCA model that was used for training.

Training process is repeated ten times to increase accuracy with two competing strategies evaluated: a lottery ticket approach [39], where the model with best

validation score is later used for testing, and an average of all ten models - results from both approaches are elaborated on in the results section. Next, the percentage of the well assumed to be drilled is increased by one percentage point and the complete training process is repeated. Increments can in practice be either shorter or longer. New models can be trained continuously and there is no underlying reason to artificially increase the intervals.

Our implementation uses TensorFlow 2.1.0 with integrated Keras library and Python 3.7. Model training was performed on Intel Core i7-8850H CPU, 32GB of RAM and NVIDIA Quadro P2000 GPU with 5 GB of GDDR5 memory providing Peak Single Precision FP32 Performance at 3 TFLOPS. Model training required 2-15 minutes (2-15 meters of drilled well at ROP of 60 m/h), depending on the simulated percentage size of the well drilled. Predictions based on the trained model are for all intents and purposes calculated instantaneously.

Hyperparameter tuning

Hyperparameter tuning is a process of adjusting various settings in the machine learning algorithm to increase its performance and is done utilizing training and validation dataset. This poses a problem as our proposed method assumes no prior access to data. Performing hyperparameter tuning on similar dataset and with the same goals can be done to overcome such issue. Such approach is utilized in other areas of machine learning, for instance a neural network detecting cats and dogs will not call for new hyperparameter tuning when detection classes are expanded to birds and rabbits since the problem at hand is technically identical from the perspective of the neural network. This is not to be confused with requirement of training a model on a similar well. This process is much more generic and likely not sensitive to geology or equipment used. Hyperparameters found to be working well for our case study are likely to provide good results when reused in model application to any bent sub directional drilling around the world. We were regrettably unable to evaluate and confirm this assumption due to lack of access to suitable dataset.

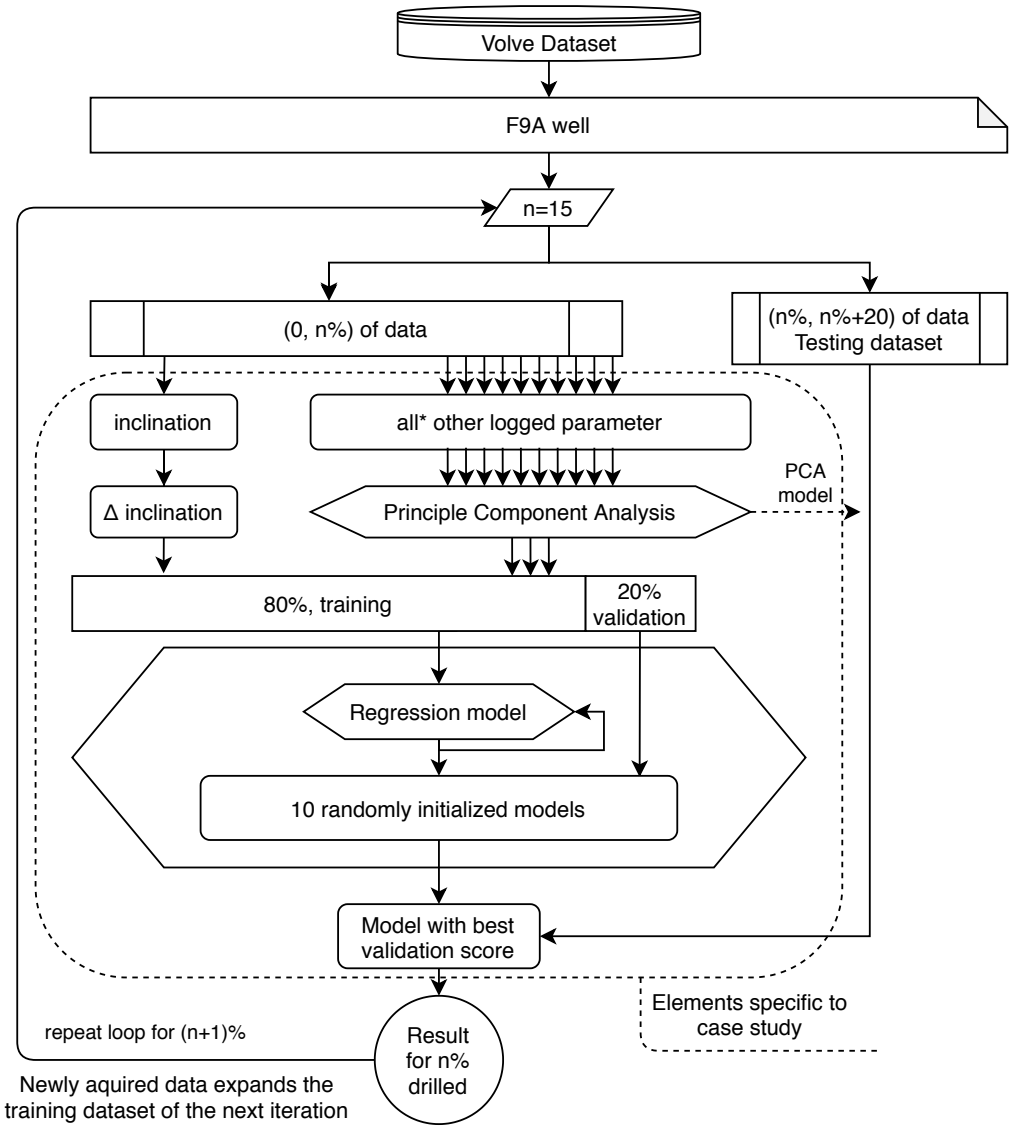


Figure E.7: Training while drilling process workflow

In our case study due to available data being limited to one well, hyperparameter tuning was performed on the same well that was later used for method evaluation. This was limited to layer size, dropout size, learning rate, kernel initialization variants, Gaussian noise levels and batch size, hence should not artificially increase the performance of evaluated case study. Hyperparameters stay constant throughout the complete drilling operation and all the iterations of the model

generated as new data becomes available.

Tuning of these parameters was done using Bayesian optimization algorithm [40]. Best parameters vary between nominal inclination and change in inclination approaches, with dropout layer at 50 percent, Gaussian noise layer at standard deviation equivalent to 0.2 percent of full scale, and approximately 350 neurons in the RNN layer and 10 and 100 neurons in final dense layer, depending on the prediction approach. Specific values can be found in the source code provided. All tuning was done with early stopping, with patience at 25 epochs and saving only the best model.

Three datapoints were selected, with 30, 55 and 80 percent of dataset used in the case study for training and validation as a basis for hyperparameter tuning exercise. Average loss of these three points was used when evaluating changing performance. Alternative methods are possible, such as evaluation based on the worst score, or evaluation based only on most difficult sections, i.e. those with little data. Method selection should be driven by specific objectives of the network under development. In our case study average overall performance was chosen as the key factor and method selected accordingly. Only three percentage points were selected to limit the time required for hyperparameter tuning, which is notorious for being time consuming. Note that PCA dimensionality reduction was not a part of final hyperparameter tuning. It was decided that this is a critical aspect of the model and therefore analysis of component quantity from 1 to 20 was performed separately, as shown before in Figure E.5.

In the future, as computational power increases, hyperparameter tuning prior to model deployment may not be necessary. As it is required to evaluate hundreds of alternative hyperparameter configurations in the tuning process, even models that are trained in mere minutes take hours to become optimized. This time has to be significantly reduced, by two orders of magnitude, to perform it during the drilling operation itself. Considering current progress in the discipline this is unlikely to happen in the next 10 years, unless new, more efficient algorithms are discovered.

E.2.4 Overfitting

Proposed method was optimized for small datasets to provide useful results as fast as possible. Small datasets are often prone to overfitting, where a machine learning algorithm *memorizes* specific datapoints instead of creating a method capable of generalizing. A number of methods were applied to tackle this problem. Typical approaches to overfitting are a dropout layer, where neurons are randomly dropped while training, Gaussian noise layer, where artificial noise is added to the signal and an architecture minimizing the number of neurons. Another approach to overfitting reduction is an ensemble of models, which is explained in detail in the results section of this paper.

E.3 Results

Results from a single sample can be visualized by plotting the past inclination data, predicted inclination data and ground-truth target values. The same method is used regardless of using nominal inclination data or incremental inclination data. This gives a good representation of the task at hand in terms of practical results that can be achieved. One sample of such chart is shown in Figure E.8. Note how the inclination prediction follows the same pattern and values relatively close to the actual data. The rotating potion of the bent sub drilling, where inclination is temporarily constant is also well represented. Note that the complete cycle of build-hold-build takes approximately 20 meters in our case study, and prediction window used is 23 meters. The y-axis refers to a local coordinate system of a sample, where first, oldest inclination datapoint is moved to zero.

There are multiple ways of describing the error between the prediction and the ground truth. It must be first established what is actually the value of interest that is being predicted, as this may change from application to application. Figure E.9a shows the mean absolute error that this paper uses to evaluate the presented model in its variations. This specific figure is a heatmap showing error for the prediction with the prediction distance on the horizontal axis and a given

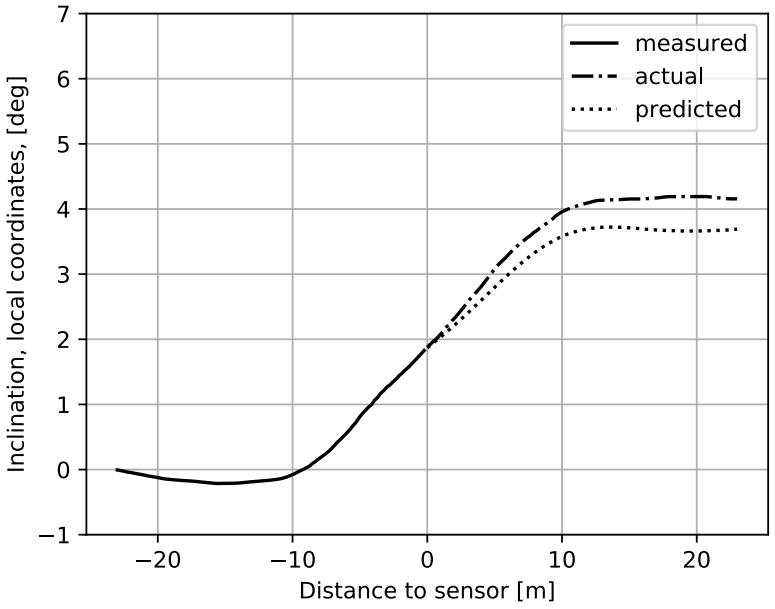
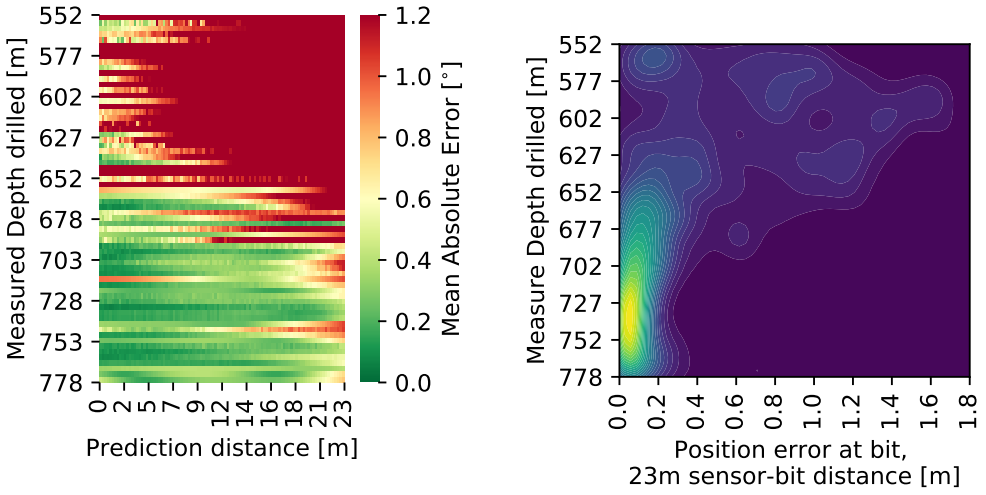


Figure E.8: Sample result, 60% of the dataset for training and validation, Measured Depth ca. 700m, incremental inclination model.

distance drilled on vertical axis. This figure immediately shows that the further from sensor one predicts, the worse the results; it also shows that predictions made early in the well are worse than the later predictions. This is in line with expectations, as at measured depth of 552 meters (after 52 meters of available data, which starts at MD=500m) the training dataset is too small to reliably predict future data yet.

Another potential way of understanding data is in terms of the positional error. Figure E.9b shows how different is the predicted bit position from the true position, considering a local coordinate system. This is calculated in the inclination plane using predicted inclination value and the fixed step of the prediction of 0.23 meters. Here, the horizontal axis shows position error, and the vertical axis measured depth drilled, making a two dimensional histogram. The position error drops all the way to and below 0.1 meters in the latter section of the dataset. Alternatively, positional confidence intervals can be plotted, as shown in Figure E.10, displaying roughly the same information, but in a more quantifiable manner. It shows both



(a) Mean Absolute Error, degrees, as a function of prediction distance.

(b) Positional error at bit, distribution heatmap.

Figure E.9: Different prediction error metrics in a function of well depth drilled, nominal method.

the median line as well as the confidence intervals, between 5th and 50th, and between 50th and 95th percentile.

Error can also be quantified in terms of R^2 value. This was calculated both based on predicted angle as well as predicted local x and y coordinates. While results for all those 3 parameters follow similar trend, they are slightly different as seen in Figure E.11a, Figure
reffig:r2x and Figure E.11c. The beginning of the well shows lack of meaningful prediction, i.e. no correlation with R^2 values below zero, which significantly improves later in the well. Note that R^2 values below zero were changed to zero for readability. The calculations were done for angle prediction at all prediction steps, while the error for coordinates were done only for the bit position, since all predictions are used to calculate the final position anyway. The prediction based on coordinates shows better results, which is likely due to averaging out of the noise in predicted data.

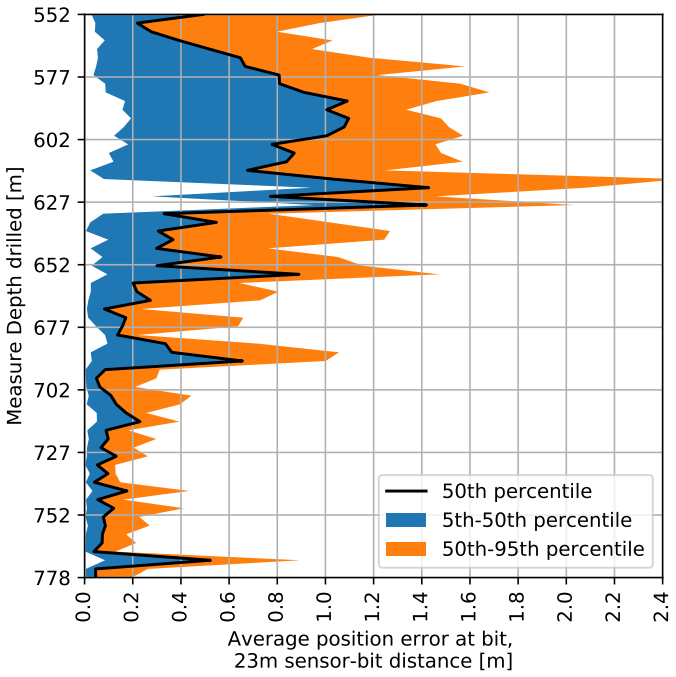
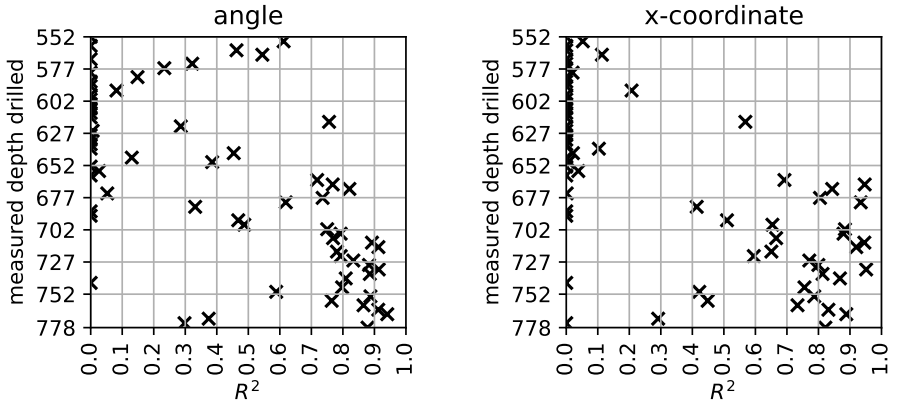


Figure E.10: Confidence intervals for bit position prediction, nominal method.

From multiple evaluation metrics we have chosen to use mean absolute error to discuss and elaborate on alternative prediction methods, as the most intuitively understood value. As the data used for case study is shared, as well as complete source code, it is possible to calculate additional metrics on demand with relative ease. It must also be stressed that presented results, in terms of accuracy, have no equivalent in current state of the art. Presented method generates values that otherwise would simply not be there; normally the inclination value between the sensor and the bit is considered simply as unknown.

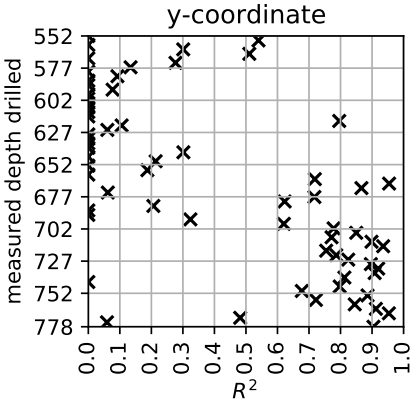
E.3.1 Alternative architectures benchmark

To highlight the benefits of including past values' information into the network two more traditional architectures were tested - MLP and extreme gradient boosting



(a) R^2 values for angle predictions, all values.

(b) R^2 values for bit position prediction, x component.



(c) R^2 values for bit position prediction, y component.

Figure E.11: R^2 scores for different predicted values.

(XGBoost) [41][3], a method that won multiple machine learning competitions. These algorithms were applied using a single datapoint per sample, without taking into account the past values. That is, the only inputs are from data strictly co-located in time with the output. Results are shown in Figure E.12. Note that in this approach the inclination change was predicted, and actual inclination

[3]Version xgboost-1.1.0 was used, Python implementation

was reconstructed through cumulative sum to calculate the mean absolute error relative to the actual inclination value.

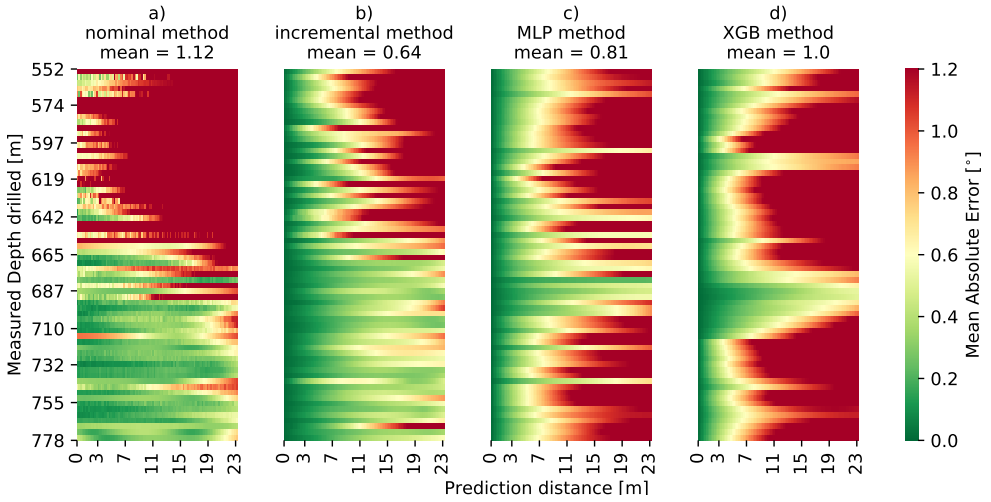


Figure E.12: Comparison between traditional and proposed approach

While both the MLP method and XGB method show mean absolute error lower than the nominal method, this is mostly due to very high error rates in the early stages of drilling. Both of the simpler methods predict inclination well throughout the well only for short prediction distance. At approximately 600 meters of measured depth drilled there is an area of low error visible, but this does not continue further into the well, suggesting area that is simply easier to predict.

Direct Comparison between XGBoost, the better performing of two simple methods, and our proposed method (incremental inclination, lottery ticket approach) was also performed, as seen in Figure E.13, where a difference between mean absolute error values are seen. What is interesting here is that the simpler model worked much better on a small dataset, although referring back to Figure E.12, error for predictions above few meters was nevertheless high. This behaviour most likely stems from the fact, that our proposed method uses a much more elaborate structure capable of finding more complex relationships. This in turn penalizes problems with small training datasets; only after collecting sufficiently large

amount of data our proposed method outperforms simpler models. XGBoost, which is an ensemble of basic algorithms, perform better on a small dataset because it requires less data to train efficiently. Recurrent neural networks require bigger training datasets to perform well, which is evident in this comparison.

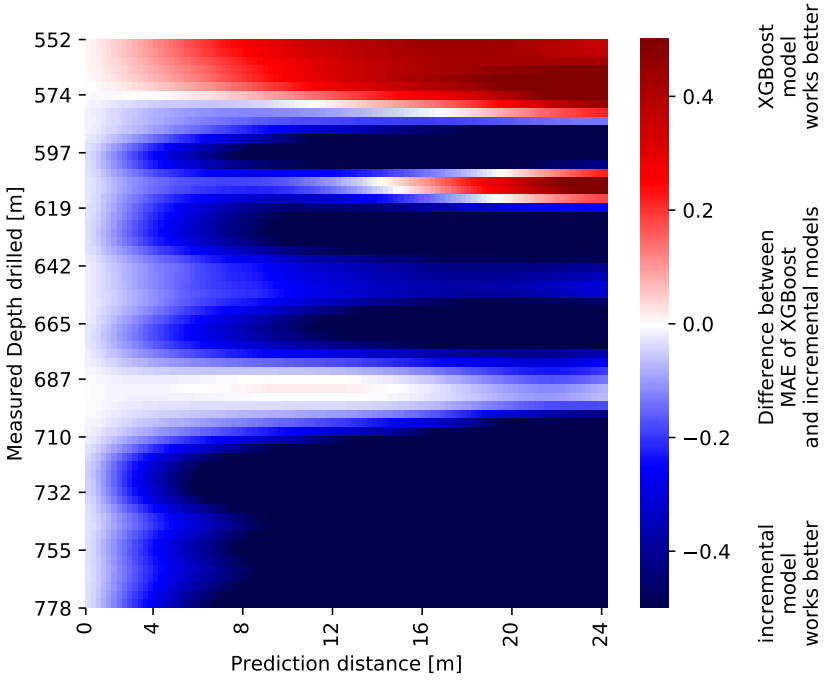


Figure E.13: Comparison between proposed approach and XGBoost

E.3.2 Incremental compared to nominal model

Although the incremental model shows significantly lower MAE scores it does not necessarily mean that it is better in all scenarios. As seen earlier in Figure E.13, nominal model performs better for further predictions. This is explored in detail in Figure E.14, where difference between the mean absolute error of two different models is calculated. Slight Gaussian blur was added to the results for easier analysis and outlier removal. With prediction distance on x-axis and drilled depth on y-axis, the color suggests which model is more accurate.

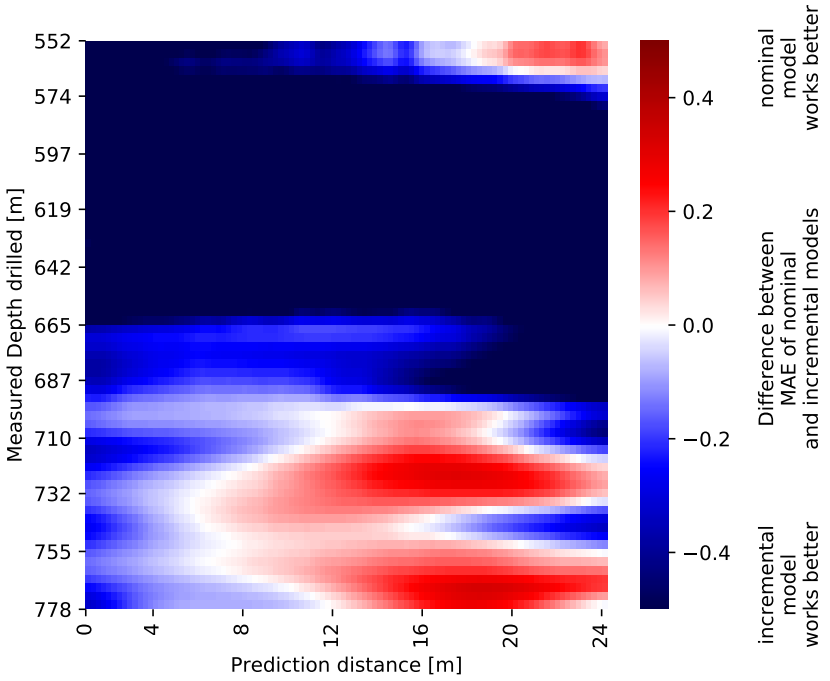


Figure E.14: Comparison between approaches

The results suggest, that if a further prediction is targeted, the nominal inclination approach is better. If however short prediction is needed, up to approximately 8 meters, it is the incremental inclination method that works best. Referring back to Figure E.16, it is clear that inclination method delivers good prediction for the first few meters from the start and provides good 10m+ predictions earlier then the alternative. Depending on requirements, one or the other approach should be selected, or potentially an ensemble of those two methods can be implemented. The difference in results comes from the target predicted value - nominal inclination or inclination change. When predicting the nominal value, the algorithm may perform poorly in the short term prediction, however for further data-points the algorithm still *aims* at the actual inclination value, while the incremental model will accumulate the errors from each consecutive prediction, as all those values are needed to recreate further inclination values.

E.3.3 Lottery ticket and ensemble results

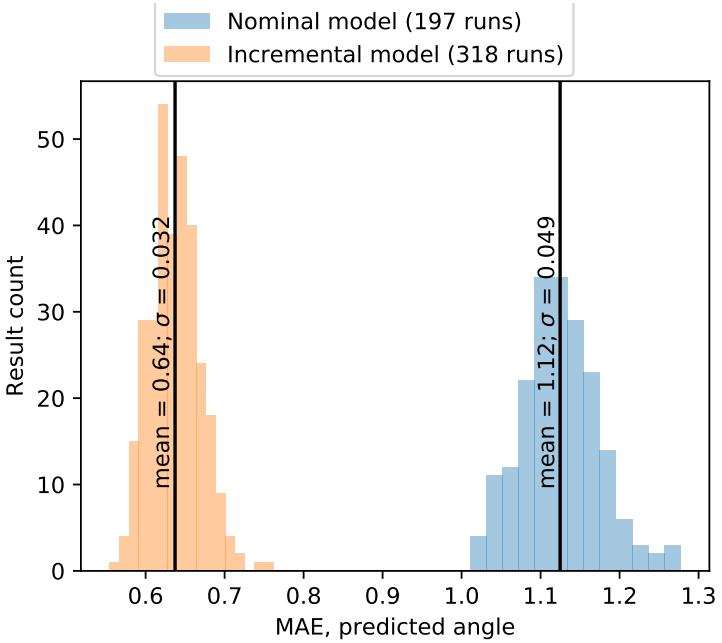


Figure E.15: Histograms of incremental and nominal method MAE, multiple runs compared.

While developing the method it became clear that the results often vary between good and bad, even if calculated for identical data, only with different random seed. This is because training process has stochastic elements, such as weights and biases initialization. Distribution of MAE values for repeated runs is presented in Figure E.15 for both the method predicting nominal inclination as well as predicting inclination change. Standard deviation at approximately 5 percent of mean MAE suggests that it is possible to improve the accuracy. This is in line with recent research discussing *lottery ticket hypothesis*, that network initialization may be simply lucky and achieve better performance [39]. Alternatively, average of models, otherwise known as ensemble, is often used to increase the prediction performance, which is especially common in climate research [42, 43]. Both approaches were evaluated by training the model 10 separate times, and in one scenario selecting the model with best validation score, and in the other taking the average prediction of all the models. Repeating model training ten times

was chosen as a balance between increased performance and increased model training time; increasing this value would continue to yield continuously smaller improvements. Additionally, predicting inclination variant and predicting of inclination change was tested, resulting in total of four different models. Results are shown in Figure E.16. When using the nominal inclination model, the mean MAE dropped from 1.12 to 1.07, approximately 5% improvement to the lottery ticket method. The inclination change method also showed 5% improvement for the lottery ticket method, with the ensemble method actually increasing MAE, although the standard distribution was significantly reduced. This suggests that the lottery ticket brings tangible, modest improvements, and should be used. There were approximately 40 simulations run for each ensemble and lottery ticket variant to find the distribution of the results.

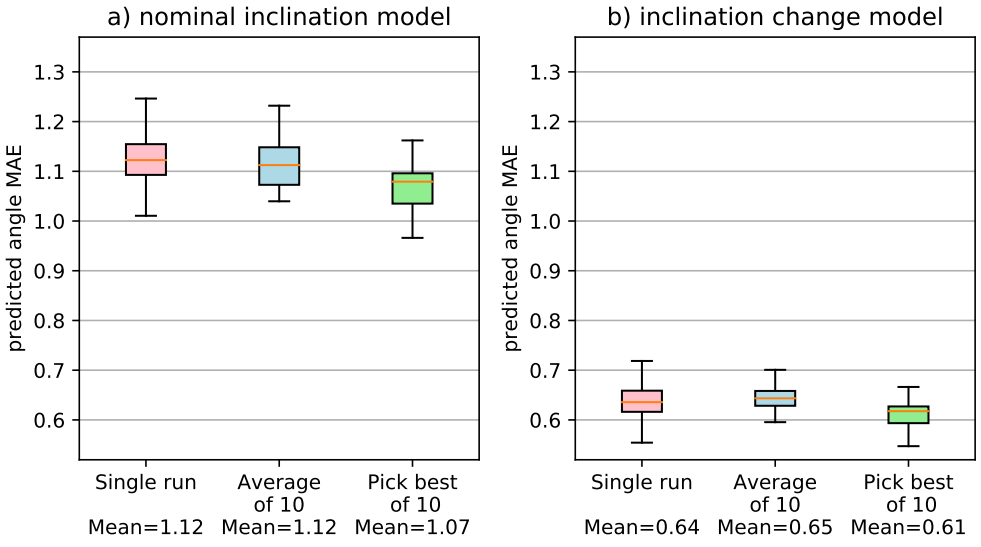


Figure E.16: Lottery ticket and ensemble performance evaluation

E.4 Discussion, usability threshold

In relation to our case study, it was possible to achieve results with mean absolute error under 0.5 degrees for prediction horizon of 23m after approximately 180m of

drilled well. This is a relatively short section of a well and an acceptable *start-up* period necessary to build a sufficient training dataset after which reasonable predictions can be made. Considering the practical use of the case study this can be considered acceptable when the target inclination is further ahead, which it typically is. In relation to other applications, such as ROP prediction, this method should be applicable as well. While ROP lacks clear sequences as in bent sub drilling, information from data directly adjacent to the prediction area undoubtedly carries useful information.

There are some caveats when this technology is deployed in the field. Considerations have to be made when changes are introduced to the bottom hole assembly. These can be minor, such as replacing a blunt bit, or major, such as change in bent sub angle. A decision has to be made whether the data from before the change should continue to be used, or if a new training while drilling model should be trained from scratch. The best solution is likely to execute both approaches simultaneously and monitor performance.

Referring to all the results presented in this paper, it is worth highlighting that the case study predicts the data gap that is 23 meters long. While various results suggest that certain models work better for a shorter prediction horizon, it still has to be considered in relation to a 23 meter long prediction model. When tasked with shorter prediction, for example in a case where a sensor is 10 meters behind the bit, results may be different.

Improvements can be achieved by utilizing logging while drilling (LWD) data, as it contains formation related information. While that information is not immediately available, and cannot be considered a real-time attribute, it certainly has potential for improved performance, as well as more efficient model training which can lead to acceptable results earlier in the drilling process.

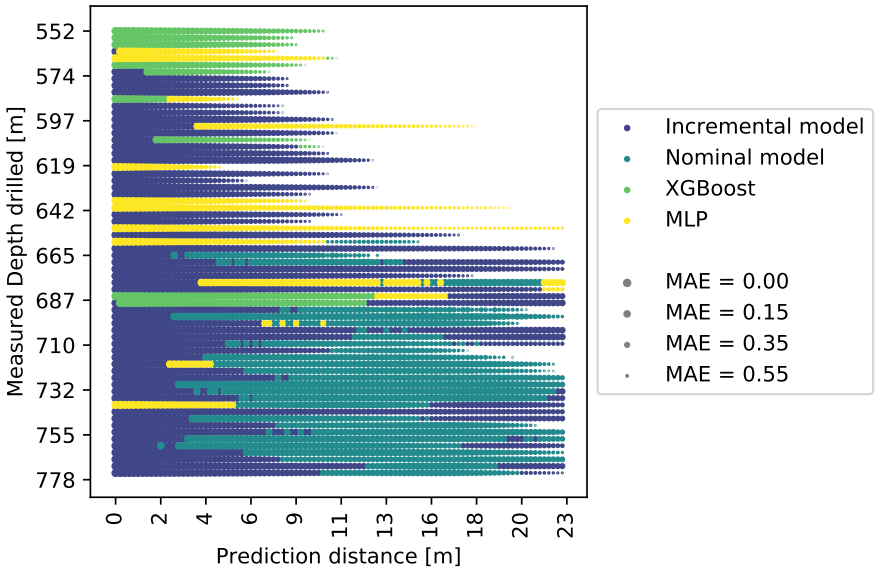


Figure E.17: Mapping best models.

E.4.1 Selecting the best model

Multiple alternative models were discussed in this paper, namely two including dynamic behaviour, nominal and incremental, and two standard regression models, MLP and XGBoost. All of these approaches seem to have strong and weak sides related to how much training is necessary and how far the prediction could be done with acceptable results. To indicate which one performs best in which area, a figure was created identifying the best out of four models.

For each point relating to specific distance drilled and prediction distance the best performing model was selected and plotted with an individual color. Additionally, areas with Mean Absolute Error above 0.6 degrees were truncated indicating that none of the explored methods worked sufficiently well. Results are shown in Figure E.17. Note that the marker size decreases with the rising error, giving an additional visual clue about the performance. The area of the chart is overwhelmingly occupied by both proposed models with dynamic behaviour, with simpler alternatives occupying very small portions of it, especially early in the

well. This again confirms previously stated conclusions, that the simpler models learn faster, but as the training set expands, the more complex ones prevail.

E.5 Conclusion

Presented method tailored for continuous learning shows good performance in the case study of predicting sensor data during directional drilling with bent motor. With existing methods being able to predict only nearest 7 meters while keeping the mean absolute error under 0.6 degrees, our proposed method achieve that goal for 23 meters of prediction most of the time. With multiple inputs decomposed to only three via PCA method, the model can be applied with little analysis in terms of available attributes, significantly reducing the workflow related to hyperparameter tuning.

Further work is needed to verify the method's applicability to predicting sensor readings of other attributes, such as gamma ray, neutron measurement, and others; and to fully quantify its potential in drilling. Presented method may also find applications in non-petroleum areas such as weather forecasting and motion capture technologies, creating models through continuous learning filling in data for failed sensors, obscured markers, and data delayed for other reasons.

Bibliography

[1] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "" Why Should i Trust You?" Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 1135–1144.

[2] Bing Liu. "Lifelong Machine Learning: A Paradigm for Continuous Learning". In: *Frontiers of Computer Science* 11.3 (2017), pp. 359–361. ISSN: 20952236. DOI: 10.1007/s11704-016-6903-6.

[3] Abdulmalek Ahmed, Abdulwahab Ali, Salaheldin Elkatatny, and Abdulazeez Abdulraheem. "New Artificial Neural Networks Model for Predicting Rate of Penetration in Deep Shale Formation". In: *Sustainability (Switzerland)* 11.22 (Nov. 2019), p. 6527. ISSN: 20711050. DOI: 10.3390/su11226527.

[4] Chiranth Hegde and K. E. Gray. "Use of Machine Learning and Data Analytics to Increase Drilling Efficiency for Nearby Wells". In: *Journal of Natural Gas Science and Engineering* 40 (2017), pp. 327–335. ISSN: 18755100. DOI: 10.1016/j.jngse.2017.02.019.

[5] Chiranth Hegde, Scott Wallace, and Ken Gray. "Using Trees, Bagging, and Random Forests to Predict Rate of Penetration during Drilling". In: *Society of Petroleum Engineers - SPE Middle East Intelligent Oil and Gas Conference and Exhibition*. Society of Petroleum Engineers, Sept. 2015. ISBN: 978-1-61399-413-9. DOI: 10.2118/176792-MS.

[6] Cesar Soares and Kenneth Gray. "Real-Time Predictive Capabilities of Analytical and Machine Learning Rate of Penetration (ROP) Models". In: *Journal of Petroleum Science and Engineering* 172 (Jan. 2019), pp. 934–959. ISSN: 09204105. DOI: 10.1016/j.petrol.2018.08.083.

[7] Chiranth Hegde and Ken Gray. "Evaluation of Coupled Machine Learning Models for Drilling Optimization". In: *Journal of Natural Gas Science and Engineering* 56 (Aug. 2018), pp. 397–407. ISSN: 18755100. DOI: 10.1016/j.jngse.2018.06.006.

[8] Mohammad Sabah, Mohsen Talebkeikhah, David A. Wood, Rasool Khosravanian, Mohammad Anemangely, and Alireza Younesi. "A Machine Learning Approach to Predict Drilling Rate Using Petrophysical and Mud Logging Data". In: *Earth Science Informatics* 12.3 (Sept. 2019), pp. 319–339. ISSN: 18650481. DOI: 10.1007/s12145-019-00381-4.

[9] Jiahang Han, Yanji Sun, and Shaoning Zhang. "A Data Driven Approach of ROP Prediction and Drilling Performance Estimation". In: *International Petroleum Technology Conference 2019, IPTC 2019* (2019). DOI: 10.2523/iptc-19430-ms.

[10] Xian Shi, Gang Liu, Xiaoling Gong, Jialin Zhang, Jian Wang, and Hongning Zhang. "An Efficient Approach for Real-Time Prediction of Rate of Penetration in Offshore Drilling". In: *Mathematical Problems in Engineering* 2016 (2016). Ed. by Cheng-Tang Wu, p. 3575380. ISSN: 1024-123X. DOI: 10.1155/2016/3575380.

[11] B. Mantha and R. Samuel. "ROP Optimization Using Artificial Intelligence Techniques with Statistical Regression Coupling". In: *Proceedings - SPE Annual Technical Conference and Exhibition*. Vol. 2016-Janua. Society of Petroleum Engineers (SPE), Sept. 2016. ISBN: 978-1-61399-463-4. DOI: 10.2118/181382-ms.

[12] Tuna Eren and Mehmet Evren Ozbayoglu. "Real Time Optimization of Drilling Parameters during Drilling Operations". In: *SPE Oil and Gas India Conference and Exhibition*. Society of Petroleum Engineers, Apr. 2010. DOI: 10.2118/129126-MS.

[13] Cesar Soares, Hugh Daigle, and Ken Gray. "Evaluation of PDC Bit ROP Models and the Effect of Rock Strength on Model Coefficients". In: *Journal of Natural Gas Science and Engineering* 34 (Aug. 2016), pp. 1225–1236. ISSN: 18755100. DOI: 10.1016/j.jngse.2016.08.012.

[14] Omogbolahan S. Ahmed, Ahmed A. Adeniran, and Ariffin Samsuri. "Computational Intelligence Based Prediction of Drilling Rate of Penetration: A Comparative Study". In: *Journal of Petroleum Science and Engineering* 172 (Jan. 2019), pp. 1–12. ISSN: 09204105. DOI: 10.1016/j.petrol.2018.09.027.

[15] Khoukhi Amar and Alarfaj Ibrahim. "Rate of Penetration Prediction and Optimization Using Advances in Artificial Neural Networks, a Comparative Study". In: *IJCCI 2012 - Proceedings of the 4th International Joint Conference on Computational Intelligence*. 2012, pp. 647–652. ISBN: 978-989-8565-33-4. DOI: 10.5220/0004172506470652.

[16] Ping Yi, Aniket Kumar, and Robello Samuel. "Real-Time Rate of Penetration Optimization Using the Shuffled Frog Leaping Algorithm (SFLA)". In: *Society of Petroleum Engineers - SPE Intelligent Energy International 2014*. Society of Petroleum Engineers (SPE), Apr. 2014, pp. 116–125. ISBN: 978-1-63266-413-6. DOI: 10.2118/167824-ms.

[17] Wanyi Jiang and Robello Samuel. "Optimization of Rate of Penetration in a Convoluted Drilling Framework Using Ant Colony Optimization". In: *SPE/IADC Drilling Conference, Proceedings*. Vol. 2016-Janua. Society of Petroleum Engineers (SPE), Mar. 2016. ISBN: 978-1-61399-401-6. DOI: 10.2118/178847-ms.

[18] Chiranth Hegde, Hugh Daigle, Harry Millwater, and Ken Gray. "Analysis of Rate of Penetration (ROP) Prediction in Drilling Using Physics-Based and Data-Driven Models". In: *Journal of Petroleum Science and Engineering* 159 (Nov. 2017), pp. 295–306. ISSN: 09204105. DOI: 10.1016/j.petrol.2017.09.020.

[19] Cesar Soares and Kenneth Gray. "Real-Time Predictive Capabilities of Analytical and Machine Learning Rate of Penetration (ROP) Models". In: *Journal of Petroleum Science and Engineering* 172 (Jan. 2019), pp. 934–959. ISSN: 09204105. DOI: 10.1016/j.petrol.2018.08.083.

[20] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Representations by Back-Propagating Errors". In: *nature* 323.6088 (1986), pp. 533–536. ISSN: 00280836. DOI: 10.1038/323533a0.

[21] Andrzej T Tunkiel, Tomasz Wiktorski, and Dan Sui. "Continuous Drilling Sensor Data Reconstruction and Prediction via Recurrent Neural Networks". In: *ASME 2020 39th International Conference on Ocean, Offshore and Arctic Engineering*. 2020. DOI: 10.1115/OMAE2020-18154.

[22] Guochang Wang, Shengxiang Long, Yiwen Ju, Cheng Huang, and Yongmin Peng. "Application of Horizontal Wells in Three-Dimensional Shale Reservoir Modeling: A Case Study of Longmaxi–Wufeng Shale in Fuling Gas Field, Sichuan Basin". In: *AAPG Bulletin* 102.11 (Nov. 2018), pp. 2333–2354. ISSN: 01491423. DOI: 10.1306/05111817144.

[23] Caineng Zou. *Unconventional Petroleum Geology*. Elsevier, 2017, p. 64. ISBN: 978-0-12-397162-3.

[24] Vitaly Koryabkin, Artyom Semenikhin, Timur Baybolov, Arseniy Gruzdev, Yuriy Simonov, Igor Chebuniaev, Maxim Karpenko, and Vasily Vasilyev. *Advanced Data-Driven Model for Drilling Bit Position and Direction Determination during Well Deepening*. Bali, Indonesia, 2019. DOI: 10.2118/196458-MS.

[25] Heng Wang. "Drilling Trajectory Prediction Model for Push-the-Bit Rotary Steerable Bottom Hole Assembly". In: *International Journal of Engineering* 30.11 (2017), pp. 1800–1806.

[26] Meishan Wang, Xiaojun Li, Ge Wang, Wenjun Huang, Yongtao Fan, Wei Luo, Jiange Zhang, Junfeng Zhang, and Xiaolei Shi. "Prediction Model of Build Rate of Push-the-Bit Rotary Steerable System". In: *Mathematical Problems in Engineering* 2020 (2020). Ed. by Francesc Pozo, p. 4673759. ISSN: 1024-123X. DOI: 10.1155/2020/4673759.

[27] Fatai Anifowose, Amar Khoukhi, and Abdulazeez Abdulraheem. "Investigating the Effect of Training–Testing Data Stratification on the Performance of Soft Computing Techniques: An Experimental Study". In: *Journal of Experimental & Theoretical Artificial Intelligence* 29.3 (2017), pp. 517–535.

[28] Fatai Anifowose, Amar Khoukhi, and Abdulazeez Abdulraheem. "Impact of Training-Testing Stratification Percentage on Artificial Intelligence Techniques: A Case Study of Porosity and Permeability Prediction". In: *5th Global Conference on Power Control and Optimization*. 2011.

[29] Khim Chhantyal, Minh Hoang, Håkon Viumdal, and Saba Mylvaganam. "Flow Rate Estimation Using Dynamic Artificial Neural Networks with Ultrasonic Level Measurements". In: *Proceedings of the 9th EUROSIM Congress on Modelling and Simulation, EUROSIM 2016, the 57th SIMS Conference on Simulation and Modelling SIMS 2016*. 142. Linköping University Electronic Press. 2018, pp. 561–567.

[30] Augustine Osarogiagbon, Somadina Muojeke, Ramachandran Venkatesan, Faisal Khan, and Paul Gillard. "A New Methodology for Kick Detection during Petroleum Drilling Using Long Short-Term Memory Recurrent Neural Network". In: *Process Safety and Environmental Protection* (2020).

[31] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3 (2014). URL: `http://arxiv.org/abs/1412.3555`.

[32] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation". In: *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference* (2014), pp. 1724–1734. DOI: `10.3115/v1/d14-1179`.

[33] Equinor. *Volve Field Data (CC BY-NC-SA 4.0)*. 2018. URL: https://www.equinor.com/en/news/14jun2018-disclosing-volve-data.html.

[34] Andrzej T Tunkiel, Tomasz Wiktorski, and Dan Sui. "Drilling Dataset Exploration, Processing and Interpretation Using Volve Field Data". In: *ASME 2020 39th International Conference on Ocean, Offshore and Arctic Engineering* (2020). DOI: 10.1115/OMAE2020-18151.

[35] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. "Scikit-Learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[36] Karl Pearson. "LIII. On Lines and Planes of Closest Fit to Systems of Points in Space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572. DOI: 10.1080/14786440109462720.

[37] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. "Artificial Neural Networks: A Tutorial". In: *Computer* 29.3 (1996), pp. 31–44.

[38] François Chollet et al. *Keras*. \url{https://keras.io}. 2015.

[39] Jonathan Frankle and Michael Carbin. *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks*. 2018. arXiv: 1803.03635.

[40] Fernando Nogueira. *{Bayesian Optimization}: Open Source Constrained Global Optimization Tool for {Python}*. URL: https://github.com/fmfn/BayesianOptimization.

[41] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Vol. 13-17-Augu. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785.

[42] James S Goerss. "Tropical Cyclone Track Forecasts Using an Ensemble of Dynamical Models". In: *Monthly Weather Review* 128.4 (2000), pp. 1187–1193.

[43] Mohammad Reza Najafi and Hamid Moradkhani. "Multi-Model Ensemble Analysis of Runoff Extremes for Climate Change Impact Assessments". In: *Journal of Hydrology* 525 (2015), pp. 352–361.

Appendix F

Data-driven sensitivity analysis of complex machine learning models: A case study of directional drilling, AT5

Tunkiel, A. T., Sui, D., & Wiktorski, T. (2020). **Data-driven sensitivity analysis of complex machine learning models: A case study of directional drilling**. Journal of Petroleum Science and Engineering, 195, 107630.

Correction: ΔS is used inconsistently in the publication: first it originally it designates a scalar, while later ΔS is a vector and Δs is used as a scalar.

Abstract

Classical sensitivity analysis of machine learning regression models is a topic sparse in literature. Most of data-driven models are complex black boxes with limited potential of extracting mathematical understanding of underlying model

self-arranged through the training algorithm. Sensitivity analysis can uncover erratic behaviour stemming from overfitting or insufficient size of the training dataset. It can also guide model evaluation and application. In this paper, our work on data-driven sensitivity analysis of complex machine learning models is presented. Rooted in one-at-a-time method it utilizes training, validation and testing datasets to cover the hyperspace of potential inputs. The method is highly scalable, it allows for sensitivity analysis of individual as well as groups of inputs. The method is not computationally expensive, scaling linearly both with the available data samples, and in relation to the quantity of inputs and outputs. Coupled with the fact that calculations are considered embarrassingly parallel, it makes the method attractive for big models. In the case study, a regression model to predict inclinations using recurrent neural network was employed to illustrate our proposed sensitivity analysis method and results.

F.1 Introduction

Sensitivity of a model describes the severity of change of the model's output related to the change of a given input value. It can give an insight in the influence of input variables on outputs. Such analysis is necessary for understanding models' behavior in terms of the change of input values, noise tolerance, data quality, internal structure, etc.

There are a number of statistical methods to evaluate sensitivity, for instance Sobol' indices [1], cobweb plots [2], various metamodels [3] and more [4]. These methods might however be not suitable for high-dimensional models of hundreds of input variables. In cases where inputs are highly dependent on each other, as in case of recurrent neural networks, sensitivity analysis remains an area of active research [4]. More recent work have introduced Shapley effect [5] to tackle this problem, however the high computational cost remains a problem.

Sensitivity information can be, to an extent, extracted by examining weights of neurons in the network. Such approach has been published in the past [6], however

it becomes impractical for networks when neurons are counted in hundreds. Most commonly however, a simple sensitivity index [7], a one-at-a-time (OAT) approach is utilized. In light of the rise in machine learning approaches applied to drilling, such as [8, 9], a rise in focus on models' sensitivity analysis is expected in the near future.

This paper presents sensitivity analysis using partial derivatives (PaD) with the dataset used for development of the machine learning model as a basis of a quasi-Monte Carlo analysis [10]. To our knowledge this is the first comprehensive exploration of PaD method of sensitivity analysis for models with number of inputs over 100; case study of directional drilling model this paper presents consists of 643 inputs and 100 outputs.

In this paper, presented case study contains hundreds of inputs and outputs and is based on the model prepared for prediction and recovery of inclination data in directional drilling [11]. The aim of this study is to explore the sensitivity analysis for complex machine learning models, such as those exploiting recurrent neural networks, where it is impossible or impractical to follow the classical methods - this is done in detail in Section F.3.2. Employing the dataset that was originally used for training, validation and testing of the model is the basis for our analysis. Basic method where an input is altered within carefully selected range and response of the output is evaluated was applied; this was in turn applied through all the samples of existing dataset as individual starting points, generating individual results. Using this method, sensitivity can be presented as a statistical distribution based on starting points representative of realistic potential inputs.

Similar work on a much smaller scale was performed by Lu et al. [12] for neural networks with 19 inputs for spool fabrication productivity studies, which was later picked up for business research [13], climatology [14], nanotechnology [15], petroleum [16], metallurgy [17] and ecology [18]. These studies were however restricted to non-recurrent models and no more than 26 inputs in total. Existing analyses lack specific focus on the base dataset, which becomes necessary with the rising number of inputs, as the dataset necessarily will not populate the complete

possible input hyperspace. This paper contributes to filling these gaps, applying and analyzing the method for big machine learning models. To our knowledge we are first in presenting, and formally defining a data-driven sensitivity analysis that is suitable for recurrent models with inputs and outputs defined as spatio-temporal sequences.

The paper is structured with focus of gradual understanding of the presented method. First, sensitivity calculations are performed using basic mathematical model as an example. Next, our novel method of data-driven approach to sensitivity is introduced. Using the same mathematical model as before, our method is applied and sample results and conclusions are presented. As a next stage, a mathematical model is replaced by a machine learning model mimicking the original one to evaluate how the results differ. This prepares the reader for analysis of our case study, a complex model for predicting continuous inclination data that utilizes 643 inputs and 100 outputs. In the case study, sensitivity analysis is performed for both individual inputs, as well as sets of inputs - a unique, new possibility in our novel method for sensitivity analysis. At the end, benchmarking against Sobol' indices is performed to evaluate how our proposed method compares with other established approaches.

To explore and better understand basic one-at-a-time method, assume that the system is described as:

$$R = f(S) \tag{F.1}$$

where R is the output and S is the input of a model described by a function f. Classical OAT sensitivity index is typically calculated at an arbitrarily selected point in the possible input domain, typically in the middle. Sensitivity index can be calculated through the following equation, that can be considered a partial derivative:

$$SI_{RS} = \frac{[R(S_0 + \Delta S) - R(S_0 - \Delta S)]}{2\Delta S} \tag{F.2}$$

Where SI_{RS} denotes sensitivity index for an output variable R per unit change in the magnitude of an input parameter S from its base value S_0. ΔS is the change applied to the input.

There are significant shortcomings to this basic approach. To illustrate we introduce a simple model of a flow rate through a nozzle:

$$Q = c \cdot A \cdot v = c \cdot \pi r^2 \cdot \sqrt{\frac{2\Delta p}{\rho}} \tag{F.3}$$

where:

- Q is Flow Rate through a nozzle, output in m^3/s.
- c is Discharge Coefficient (unitless), $c \in (0.5, 1)$
- A is Nozzle Area in mm^2
- v is Flow Velocity, in mm/s
- r is Nozzle Diameter, in mm, $r \in (1, 50)$
- Δp is Pressure across the nozzle, in Pa, $\Delta p \in (0, 1000000)$
- ρ is Fluid Density, in kg/m^3, $\rho \in (700, 1300)$

By using a spider plot, as seen in Figure F.1, it is visualized how the output value will change per percentage change in input. All inputs are set at their median value, considering their potential range, i.e. for discharge coefficient c the value is set at $(0.5 + 1)/2 = 0.75$. Keeping all other values at their median value, the output Q can be plotted as a function of c from -33.3 percent $(0.75 - 0.75 * 33.3\% = 0.5)$ to +33.3 percent $(0.75 + 0.75 * 33.3\% = 1)$. This is done in the same manner for all the inputs.

There is a lot of information that is hidden on this plot. For example, the change in *nozzle diameter* r suggests that the *flow rate* can be adjusted by this parameter between $0m^3/s$ and $0.05m^3/s$. This is however only true for the selected middle

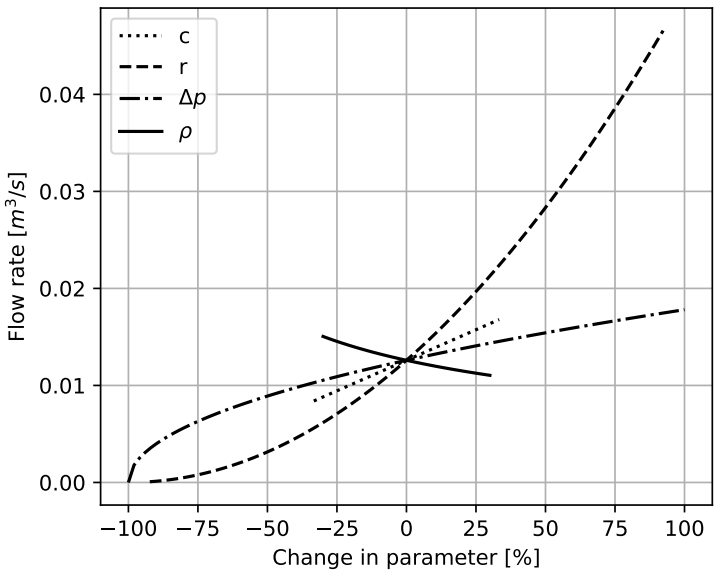


Figure F.1: Sample Spider plot of mathematical model's sensitivity.

values for all the other parameters. More general statement can be made that doubling the *nozzle radius* will quadruple the *flow rate*, however the chart does not uncover that this is not valid when the pressure drop Δp is equal to zero. Most importantly however, OAT methods cannot cover a significant part of the high-dimensional Euclidean space [19]. The significance of this grows with the number of dimensions, which is especially high in case of recurrent neural networks.

F.2 Data-driven approach to sensitivity of data-driven models.

In machine learning problems related to sequential data Recurrent Neural Networks (RNN) are often utilized [20]. Abstracting from the specific internal structure of the network, the model takes multiple inputs, in the same order as they exist in a dataset. Model that will take into account last n values of a single

input will technically have n inputs. This custom length will be driven by the performance of a given model as well as computational cost of training.

Neural Networks can have thousands of inputs without reaching a bottleneck in terms of computational power when training the model. Areas where machine learning often excels are commonly both highly dimensional with interdependent inputs, such as RNN and other complex models, which makes them unsuitable for classical sensitivity analysis methods. This paper proposes a number of approaches that were specifically designed for data-driven models with high number of inputs and outputs, addressing both of these issues.

A number of assumptions are made in the proposed method that stem from specific nature of data-driven models. This enables simple calculations and it leapfrogs shortcomings of the classical methods. The basis for proposed sensitivity analysis is the training, validation, and testing dataset of the model. It is a requirement that this dataset represents the complete range of possible inputs and outputs relatively well. While this may be considered too strict or unrealistic requirement, by definition a data-driven model is applicable to inputs and outputs that are covered by the training material. Data-driven sensitivity analysis will be valid only for ranges of values represented by the dataset at hand, the same way as the data-driven model is valid only for the input data from the mathematical vicinity of the training set. It has to be stressed, that sensitivity analysis in any form is connected to the range of inputs and, additionally in our case, the distribution of the samples.

Typical training/validation/testing split of the dataset is removed in our method and complete dataset is used to maximize the amount of samples and therefore coverage of the inputs' hyperspace. While developing a data-driven model this split is introduced to ensure that the model can generalize and not simply *memorize* the provided samples. In contrast, our method utilizes the data as a set of representative and realistic inputs to the model, and therefore the split is no longer necessary.

Limiting calculations to the data points existing in the dataset, instead of all possible combinations, apart from significant reduction in calculation time, avoids diluting results with impossible inputs. Our case study consists of multiple sequential data inputs organized along the increasing depth, for example inclination change of the well or surface torque. These data are continuous and never rapidly oscillate between maximum and minimum value, only smoothly rises at different rates. Checking sensitivity for unrealistic scenarios is non-productive, since the model is not valid for them.

Second assumption is that the existing dataset is balanced in terms of probability of inputs. That is, if a given model predicts daily mean temperature, the number of samples should be distributed evenly across all twelve months. This is done to ensure that any data skew that may exist in the dataset does not translate to a skew in sensitivity analysis. Additionally, an artificial limitation can be introduced to a dataset, for example to explore sensitivity related to data exclusive to February.

If a dataset is imbalanced, methods commonly used for dealing with such problem in the domain of machine learning can be applied. Undersampling the over-represented data can be performed, where some samples are simply removed. Duplicating underrepresented samples can also be done, however this should be done with caution, so that the dataset does not become overly skewed towards specific data. For example, if there are 1000 samples for weather data for each month except for February, duplication would be acceptable if it extends the sample size from 750 to 1000. On the other hand, should February have only 100 data points, such process would be questionable.

Presented SA methods, while are rooted in one-at-a-time methods, are applied to an existing dataset populating the complete hyper-space of probable inputs. This means that the computational complexity in big-O notation is $O(n \cdot m)$ with n being the sample count and m being the input count. The amount of outputs does not influence the computational complexity. Our case study consisting of 643 inputs, 100 outputs, and with a dataset of 1300 samples required just over 1

hours to calculate all the results[1]. Our method is also considered embarrassingly parallel, i.e. it is trivial to perform calculations on multiple computers at the same time. Therefore, it can be easily implemented in frameworks such as Hadoop or Spark should the data size require that.

The result of the presented methods is a probability distribution. One-at-a-time methods are applied to the many samples of the dataset generating a set of individual results that, as a whole, present global sensitivity of the model.

F.2.1 Method introduction, flow through a nozzle

Basics of the method are presented using the example of flow rate through a nozzle shown earlier in Equation F.2 before exploring an oilfield related case study utilizing recurrent neural networks. Dataset was generated for inputs for this model containing 1000 samples with normal distribution at the center of the range indicated for Equation F.3 and standard deviation equal to 0.2 of total range width. For example, samples for *density*, ranging from 700 to 1300 kg/m^3, were randomly generated using normal distribution with mean value of 1000 kg/m^3 and standard deviation of 120 kg/m^3. Note that this is done for illustration purposes only in-lieu of real dataset. Systems in practice rarely follow pure Gaussian distribution hence using true dataset is important.

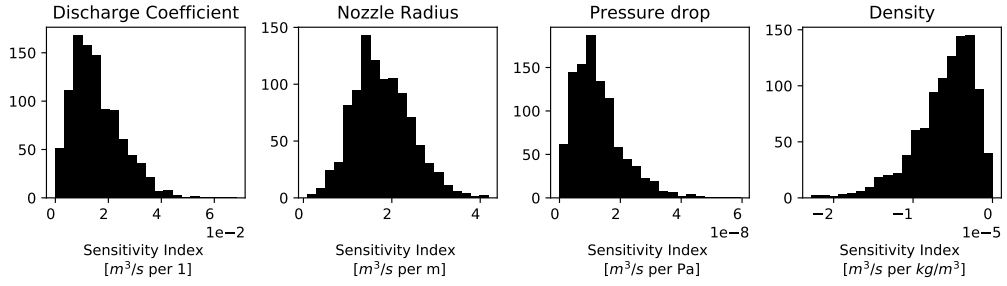


Figure F.2: Sensitivity index, histogram.

[1]Calculations were performed on Intel Core i7-8850H, 32GB RAM, Python 3.6.9, SALib library in version 1.3.8.

Sensitivity Indices were calculated using Equation F.2 with previously generated sample data as a starting point. Value of ΔS was set to 0.1; the influence of the value selection is explored further in the paper. The following calculations were performed for each of the four inputs to gauge their respective sensitivities. Figure F.2 shows the distribution of sensitivity indexes over the dataset. To better understand the data behind this Figure F.3 is also plotted, where sensitivity index is shown as a function of the input value. What may be surprising is that it is not a continuous line. This is because sensitivity index, as described by Equation F.2, deals with absolute values. For example, when pressure across the nozzle is high, small increase in its diameter will cause a large nominal flow increase. Alternatively, at very low pressures, even big nozzle increase will not yield significant change.

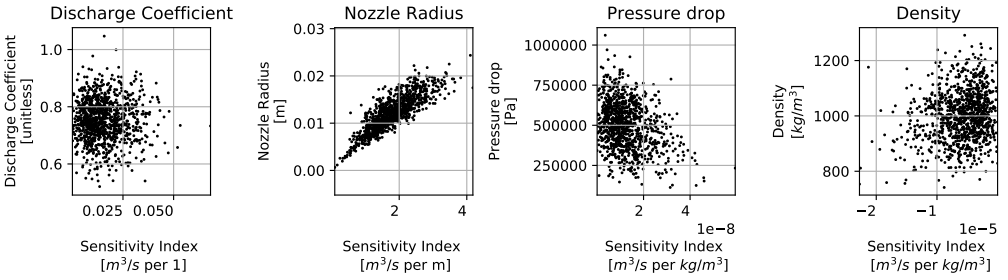


Figure F.3: Sensitivity index as scatter plot of dataset.

To check the relative impact on the output one may calculate sensitivity divided by the model output at nominal input value, which this paper refers to as relative sensitivity index:

$$RSI_{RS} = \frac{[R(S_0 + \Delta S) - R(S_0 - \Delta S)]}{2\Delta S} \cdot \frac{1}{R(S_0)} \tag{F.4}$$

This will produce a relation seen in Figure F.4, where a fixed change in nozzle size will affect relative flow output more at smaller diameters, i.e. change from 1.5mm nozzle to 2.5mm, and less for bigger nozzles, i.e. from 20mm to 21mm

nozzle. Since this response is disconnected from the absolute value of the output, it produces a continuous line.

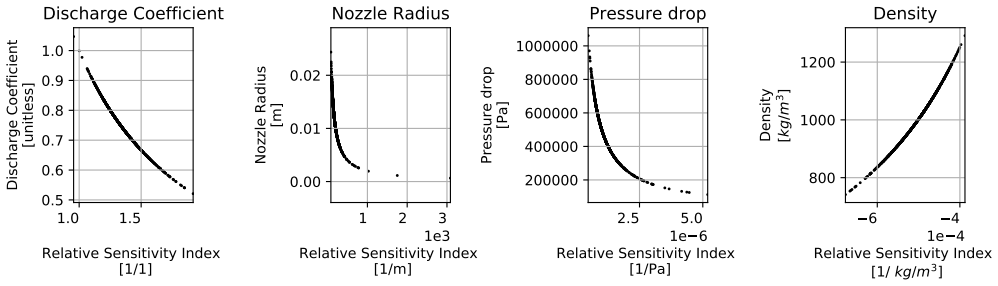


Figure F.4: Relative sensitivity index.

One can further explore the system response with Elasticity Index, or elasticity of a function [21], given by Equation F.5 below.

$$EI_{RS} = \frac{[R(S_0 + \Delta S) - R(S_0 - \Delta S)]}{2\Delta S} \cdot \frac{S_0}{R(S_0)} \tag{F.5}$$

The equation was again applied to all the samples and all the inputs, with results plotted in Figure F.5. Elasticity Index in practice uncovers the exponent of the function, hence in case of Equation F.3 it is equal to 2. Derivation of this relationship is presented in the Appendix.

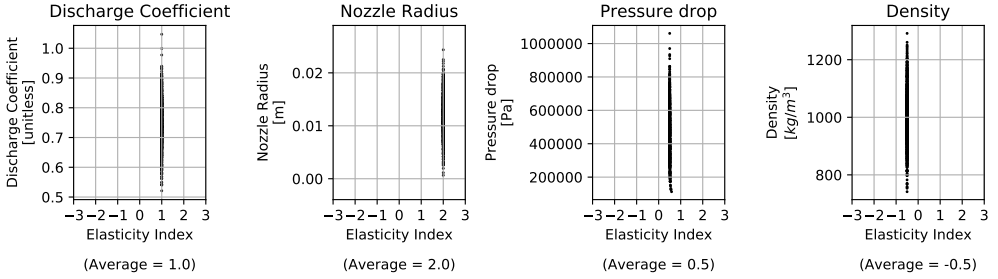


Figure F.5: Elasticity index

When exploring the produced figures one can see a number of issues with the calculated data. First and foremost, the scale of the Sensitivity Index values

varies by orders of magnitude. For *nozzle radius* it is between 0 and 4, and for *pressure drop*, intuitively a very important parameter for flow through a nozzle, it is in range of 10^{-8}. Note that the units themselves are different ($m^3/s/m$, $m^3/s/Pa$, and others) which mean that the values cannot be compared.

While in the case of a simple equation the relative sensitivity index and elasticity index bring interesting additional information, in our case study focus will be on the basic sensitivity index. Nevertheless, it is beneficial, for the sake of presentation of the method, that all three equations will be explored. They are a natural next step, and are needed to understand why the simpler, and seemingly less informative equation is used.

F.2.2 Practical problem of scale

One problem that was identified while inspecting results produced in the previous section is units and scales. Referring back to Figure F.2 and inspecting the x-axis it is clear that there is no straightforward information about which input has high sensitivity compared to others. This is the key information that sensitivity analysis is seeking to uncover when probing a given model.

It is worth reiterating that machine learning models exist with a strong relationship to a dataset of samples, and in consequence, to the range in which inputs exist. Looking back at the sample nozzle flow Equation F.3, while mathematically sensitivity is defined by the equation alone, the application makes it tied to the range of inputs. For example, in practical terms, sensitivity of the *nozzle radius* input can be considered high if it is ranging from 1 to 50mm, but it will be very low if it is bound between 5mm and 5.1mm.

Nozzle Equation F.3 was encapsulated in a function such that all inputs are scaled between 0 and 1 in relation to generated samples. Output is also scaled between 0 and 1 for values generated from the samples. While this looks uncanny for a mathematical equation this is a very typical approach for a machine learning

model. Histograms seen in Figure F.6 were generated the same way as described in the subsection before, and therefore are analogous to Figure F.2.

The scaling process did not affect the shape of the histograms in a significant way, but the sensitivity factor can now be evaluated in terms of expected range of inputs and outputs. For example, one can see that when radius input is changed by 10 percent of full scale, output can be expected to also change by 10 percent, since the most common sensitivity value on the histogram is 1. In case of *discharge coefficient* sensitivity is smaller. 10 percent change will typically affect the output by 2.5 percent of expected full scale (one to 0.25). This can be tied to the spider plot back in Figure F.1, where the model response is presented. Linearly approximating the curves one can roughly see the same values as presented here - the steeper the curve the higher the sensitivity index. Note that this is a distribution of results, hence the response discussed above should be considered mean sensitivity of the model.

After scaling, inputs and outputs should be considered unitless with a scaler attached to them that describes the scaling parameters, i.e. original minimum and maximum values. Therefore the sensitivity index calculated with scaled data becomes unitless as well. Alternatively, scaled values can be considered percentage of range, making the unit of sensitivity index to be a percentage of range of output per percentage of range of input. This method was used before in relation to sensitivity analysis of neural networks [12] highlighting the same problems related to units and scale.

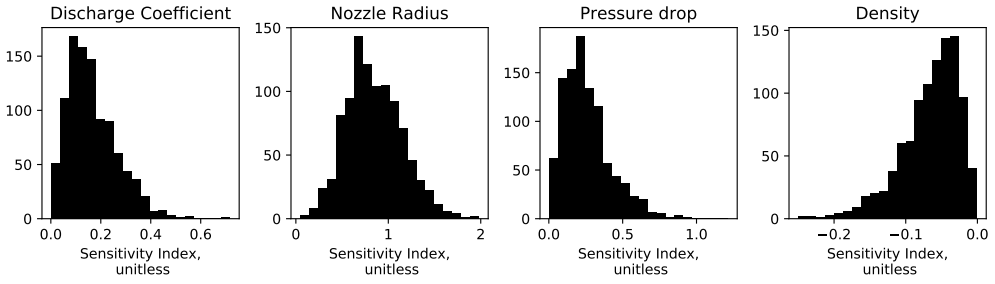


Figure F.6: Sensitivity index, scaled function, histogram.

Analyzing scatter plots in Figure F.27 showing relationship between the sample value and sensitivity index there are no immediate differences due to scaling of the dataset. For brevity this figure is reproduced in the appendix.

Moving to Relative Sensitivity Index in Figure F.7 and comparing to non-scaled charts seen previously in Figure F.4 one can notice that the result is no longer a continuous line. There is some disturbance introduced due to scaling and this effect is even stronger when inspecting Elasticity Index in Figure F.8, where EI is strongly pulled towards zero at inputs close to zero. The average still indicates what kind of function is behind the model, but it is no longer precise. as indicated by the average values below the plots.

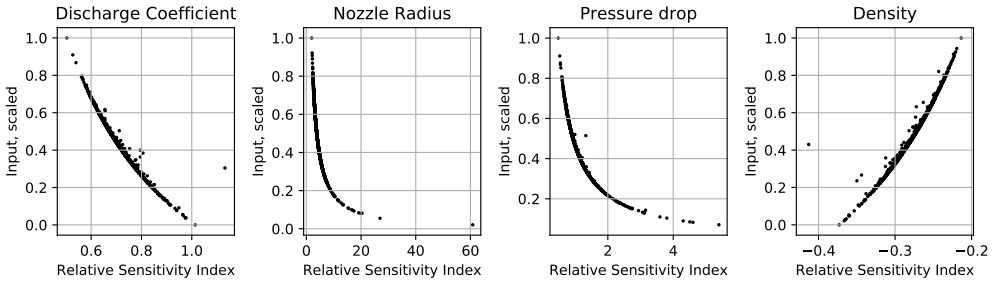


Figure F.7: Relative Sensitivity index, scaled function, scatterplot.

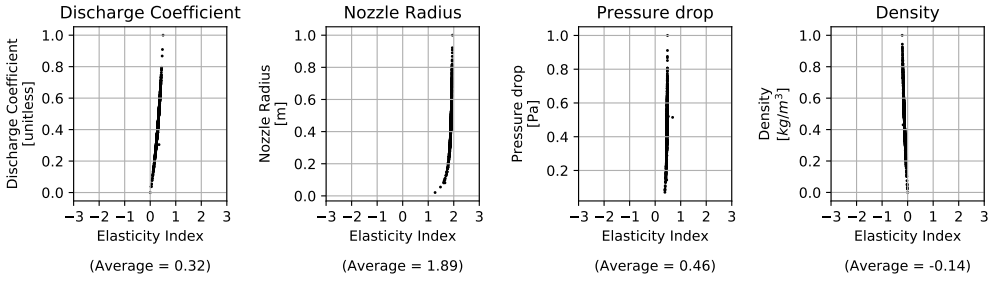


Figure F.8: Elasticity index, scaled function, scatterplot.

Analysis of results of the scaled inputs and outputs is easier, as it can be done in relation to the full scale of these inputs and outputs. It is easy to see that sensitivity to the *nozzle radius* is highest, followed by *pressure drop*, *discharge*

coefficient and with *density* being the lowest, which additionally is identified as inversely correlated with the output.

F.2.3 Applying method to a machine learning model

While sensitivity index and the other methods used provide clear insights to the inner working of a model when applied to simple equations, a study was done in relation to machine learning models. To evaluate method's suitability for such scenario, an ML model of the previously used flow through a nozzle equation was created using a Random Forest regression model [22] made with previously generated samples. Common best practices were used while making the model, including a train/test dataset split (here 50/50 split) as well as basic hyperparameter tuning to find the optimal amount of estimators and forest depth[2]. Model results of the testing portion of the dataset evaluated against the equation output have an R^2 value of 0.976. This can be considered a very good fit for a machine learning model and can be considered representative of performance of the best case real-world ML models.

Scaling of inputs and outputs in the range of*(0,1)* was also applied. This is considered standard practice when creating ML models, although not strictly necessary for the algorithm used here. The transformation was nevertheless applied, since our method of evaluating sensitivity works best on a scaled model. Note that there is no practical reason to create an ML model of a known equation and here it is done only for the purpose of evaluating the method.

Same type of analysis was performed on the ML model as it was on the mathematical model to evaluate how the intrinsic inaccuracies of the ML model influence the results. Figures F.9 and F.10 showing results for the ML model correspond to Figures F.2 and F.3 produced for the equation-based model. Most important takeaway is that the histograms of sensitivity indices for both models show approximately the same shapes and values, while the scatter plots are significantly different. When calculated for mathematical model, scatter plots for all but one

[2]identified values are: n_estimators = 93, max_depth = 3

variables seen in Figure F.3, showed approximately Gaussian distribution along both axes. For the ML model, Figure F.10, scatter plots are significantly different taking various shapes that were not present before.

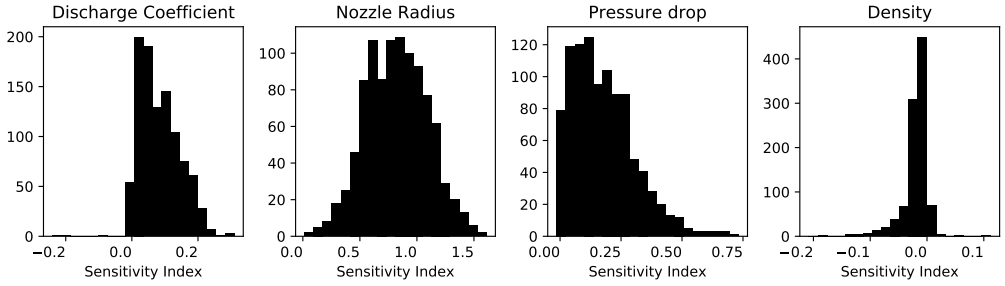


Figure F.9: ML model Sensitivity Index histogram

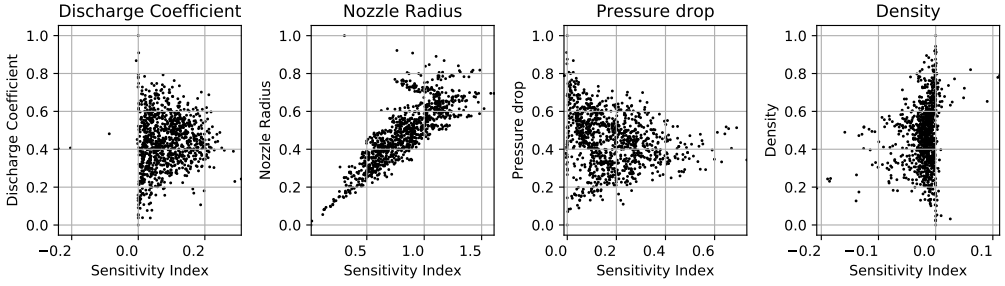


Figure F.10: ML model, scatter plot of Sensitivity Index.

Relative Sensitivity Index was also calculated for the ML model, seen in Figure F.28; same charts for mathematical model are in Figure F.4. Clear smooth lines are no longer visible, except for the the *nozzle radius*. For other three parameters it is hard to identify even the direction of the original curve. This means that Relative Sensitivity Index is very susceptible to the inaccuracies even in a case of very accurate ML model and its usefulness will be limited for data-driven models. Lastly, the Elasticity Index was analyzed, as seen in Figure F.29; mathematical model-based results are in Figure F.5. Average values became closer to zero than in the mathematical model with scaling and data is overall difficult to interpret. Conclusions related to the exponent of the equation used can no longer be drawn.

The conclusion from ML model analysis is that our proposed method focuses mainly on the distribution of the sensitivity index. Calculating additional metrics, such as relative sensitivity index and elasticity index in our experience did not bring forward any further valuable information.

F.2.4 Uneven distribution

Sensitivity analysis is closely tied not only to the range of inputs, but also to their distribution. To uncover it, another simulation was performed with the nozzle flow equation, but this time the distribution of the *discharge coefficient* was changed, with 800 samples generated at fixed value of 0.5 and 200 samples generated at a fixed value of 1, making a strongly bi-modal distribution. This does not change the range of the inputs, but significantly alters how the model behaves on average.

For a direct comparison *(0,1)* scaling was performed and machine learning model created. Analysis of histograms and scatter plots (Figure F.30 and A.F.31, appendix) shows reduced sensitivity of all inputs. This is caused by overrepresentation of samples with low *discharge coefficient* and therefore low flow rate, and in return - low sensitivity in terms of percentage of output range. This situation can only be uncovered via analysis on the inputs' distribution, and later by analyzing sensitivity for both clusters separately. This is outside of the scope of this study.

F.2.5 Summary of the method

Different aspects of dataset-based sensitivity analysis were evaluated on a synthetic example in the previous sections. Our conclusion is that the most condensed information is retained within median value and distribution of Sensitivity Index when calculated on a *(0,1)* scaled values. Coincidentally, scaling of values, both inputs and outputs, is a standard practice in machine learning, making application of the method very simple.

Relative sensitivity index was found to not bring significant additional information when exploring even our very accurate machine learning model. Furthermore, this parameter is best evaluated on a scatter plot, what becomes impractical when hundreds of inputs are being analysed. Elasticity Index can potentially bring information about the type of equation behind the model, it is however sensitive to scaling and can suggest misleading values. Additionally, it cannot be used when probing multiple inputs, which is useful in case of recurrent neural networks, as presented in further chapters.

F.2.6 Pseudocode

To aid in the implementation of the presented method pseudocode is provided in Algorithm F.1 demonstrating calculation of Data-Driven Sensitivity Index. Note that the output of the algorithm is a 3 dimensional array that contains all local sensitivity indices for all samples, inputs, and outputs. One can visualize this data in a number of ways, for example calculating median sensitivity of individual inputs by calculating it over samples and outputs such as in Figure F.17. In this case, inputs along one channel were probed individually.

If a particular input is of interest, median sensitivity on outputs can be calculated along samples on that one input, such as in Figure F.18, where one particular input in the inclination channel is investigated. It is possible to visualize data as a heatmap, where median sensitivity is calculated of all samples per individual inputs and outputs, resulting in a 2D map of sensitivity between inputs and outputs, as seen in Figures F.20 and F.21. If samples are sorted by a meaningful dimension, depth, time, distance, etc., it may be useful to calculate mean of all inputs, retaining the sample and input dimension, to investigate the distribution of sensitivity along that meaningful dimension. In our case study, such analysis was performed and is presented as a 2D map in Figure F.22.

Pseudocode for calculating sensitivity index per complete channel is provided in Algorithm F.2. It is particularly useful in relation to RNNs, where inputs are often given in a series along depth or along time. It can also be suitable when

Algorithm F.1: Data-Driven Sensitivity Index (DDSI), per single input

Data: X := [X_1,X_2, ... , X_n]
X_n := [$x_{n,1}$, $x_{n,2}$, ... , $x_{n,m}$]]
an array of n *samples with* m *inputs*
Δ typically *0.1, or 10% of input range*

Result: DDSI := [SI_1, SI_2, ..., SI_n]
SI_n := [$SI_{n,1}$, $SI_{n,2}$, ..., $SI_{n,m}$]
$SI_{n,m}$:= [$si_{n,m,1}$, $si_{n,m,2}$,..., $si_{n,m,w}$]
an array of individual, local Sensitivity Indices, through n *samples,* m *inputs, and* w *outputs.*

```
1  Function Model([x_{n_i,1}, x_{n_i,2}, ... , x_{n_i,m}]):
2      model calculations;
3      return [y1,y2, ... ,yw] ;
4  for i:=1 to n do
5      for j:=1 to m do
6          samplePlus = X[i];
7          samplePlus[j] += Δ;
8          outputPlus = model(samplePlus);
9          sampleMinus = X[i];
10         sampleMinus[j] -= Δ;
11         outputMinus = model(sampleMinus);
12         oneSensitivity = (outputPlus - outputMinus) / 2Δ ;
13         DDSI[i,j] = oneSensitivity;
```

inputs may be grouped together for other reasons, for example in weather models sensors can be grouped into temperature, barometric, and rainfall channels.

The output of the algorithm is a 3 dimensional array, with dimension for samples, channels, and outputs. Visualizing one channel at a time, one can visualize distribution of channel's sensitivity via 2D heatmap, as in Figure F.13, or potentially easier to read median and percentile statistics reducing the sample dimension, as seen in Figure F.14.

F.3 Case study model - inclination in directional drilling

F.3.1 Introduction of the model

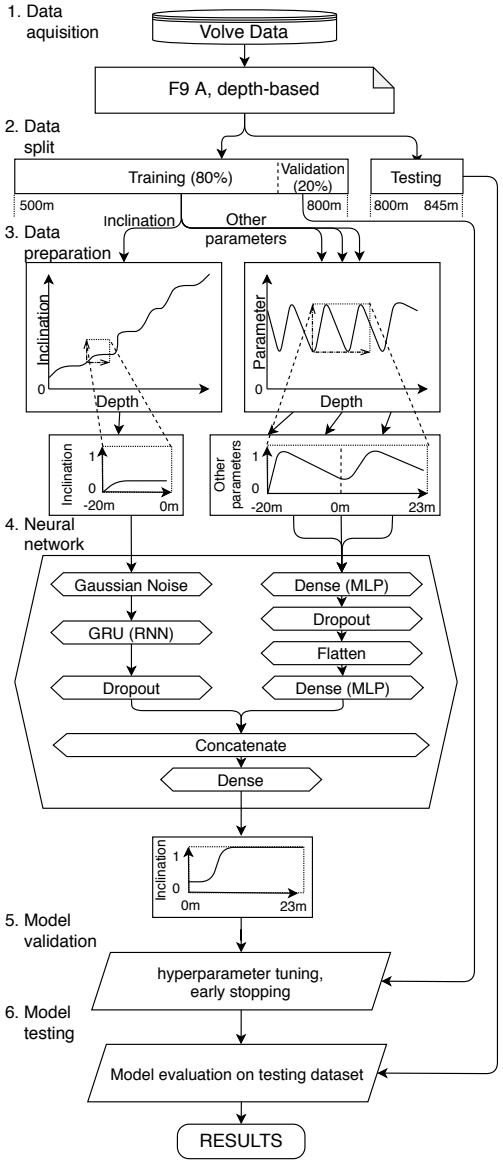


Figure F.11: Case study model flow chart.

To evaluate proposed methods of sensitivity analysis a case study machine learning model was employed which was previously used for prediction of continuous inclination data in directional drilling [11]; data and source code is available on GitHub [23]. This model predicts inclination values that are artificially made lagging 23 meters behind the bit. It is a mixed approach of trend prediction as well as regression by correlation with other parameters assumed available real-time. Flowchart from aforementioned paper showing simplified model structure is shown in Figure F.11.

Data were taken from the Volve dataset open sourced by Equinor [24], which covers an offshore oil field off the coast of Norway. Part of a well F9 A, starting from 500m to 845m measured depth was selected as containing the longest directional drilling section performed with a bent motor. Three parameters assumed available all the time were selected in addition to the past inclination data: *Surface Torque*, *Rotary Speed* and *Rate of Penetration*. Data were split into training and validation, 500m to 800m, with 80/20 split, and testing, 800m to 845m, which were used only for final score calculation.

All inputs and outputs of the model are scaled between 0 and 1, and all calculations presented in this paper are done in relation to those scaled values. Additional normalization step was introduced to the inclination input channel. To normalize the samples in relation to absolute inclination, a local coordinate system was employed. This way, inclination input channel samples always start at zero, see Figure F.12 for reference. Global scaling of the inclination was selected such, that after introducing the local coordinate system the maximum value of inputs in the inclination channel should be maximum 1.

The model contains a branched neural network, with one branch with Gated Recurrent Unit [25] RNN taking 85 inputs, the inclination data, and a second branch with multi-layer perceptron (MLP) taking 3x186 inputs from the additional available parameters. Inputs and outputs are considered in terms of relative position from the introduced start of the gap. The RNN branch receives inclination values from position -20 meters to zero meters relative to the gap. The MLP branch receives values from three different measurements, the *Rotary Speed*,

Surface Torque, and *Rate of Penetration*. That data are input from position of -20 meters to +23 meters relative to the gap. The model returns 100 output values as prediction of individual inclination values at location from 0 meters to +23 meters.

One can consider this model as utilizing four input channels and one output channel. This way the model can be reduced, for the purposes of the analysis, to four separate vector inputs and one vector output. Example results from the case study model can be seen in Figure F.12. In the top subplot dashed line denotes inclination data input, while the solid line is a prediction, and the dotted line is ground truth. The other three plots present the surface parameters that are considered available all the time. Note that the output is a continuous line. This result is not forced through algorithm, but achieved purely through neural network training.

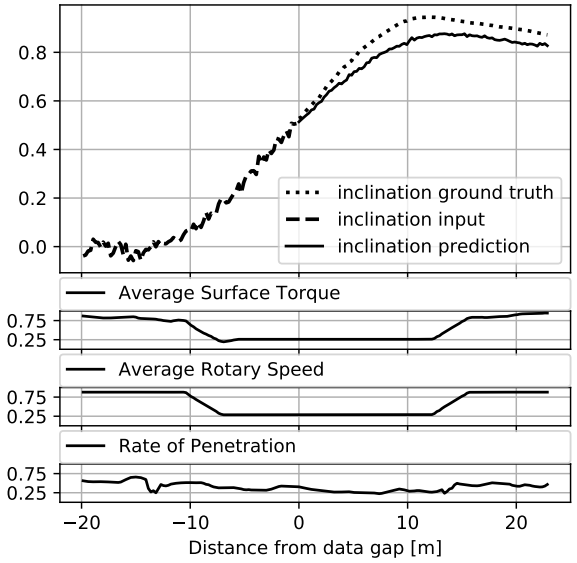


Figure F.12: Input and output sample from the case study model.

For the purpose of the sensitivity analysis the case study model was considered as follows:

$$Y = f(X) \tag{F.6}$$

To reflect the structure of the input consisting of four distinct channels, further expansion to $X_t = f(X_t^A, X_t^B, X_t^C, X_t^D)$ is made, where: X^n denotes the n^{th} input channel. Let's consider the model to be:

$$Y_t = f(X_t^A, X_t^B, X_t^C, X_t^D) \tag{F.7}$$

where

$$Y_t = \left[y_{t-m_0|t}, \ \cdots, y_{t-1|t}, y_{t|t} \right]^T \tag{F.8}$$

$$X_t^A = \left[x^A_{t-m_1|t}, \ \cdots, x^A_{t-m_0-2|t}, x^A_{t-m_0-1|t} \right]^T \tag{F.9}$$

$$X_t^n = \left[x^n_{t-m_1|t}, \ \cdots, x^n_{t-1|t}, x^n_{t|t} \right]^T, \quad n = B, C, D \tag{F.10}$$

and $m_0 = 100$ and $m_1 = 186$, $y \in R$ and $x \in R$.

F.3.2 Sensitivity index of input channels

To analyze the model top to bottom, sensitivity of the complete channel is explored first, i.e. related to multiple inputs at the same time. This is done the following way:

$$SI_t^A = \frac{f(X_t^A + \Delta S, X_t^B, X_t^C, X_t^D) - f(X_t^A - \Delta S, X_t^B, X_t^C, X_t^D)}{2\Delta S} \tag{F.11}$$

where SI_t^A is sensitivity of the channel A, and

$$\Delta S = \begin{bmatrix} \Delta s, & \cdots, \Delta s, \Delta s \end{bmatrix}^T, \Delta s = 0.1 \tag{F.12}$$

This generates multidimensional results and there are many ways to visualize them. The model is predicting 100 continuous data points (23 meters), hence they are plotted along x axis as a distance to the start of data gap. Complete results can be presented as a heatmap, as seen in Figure F.13, however this makes quantitative analysis difficult. It is clear that sensitivity is approximately 0.9 at distance to data gap 0 (bright yellow area) and becomes more spread out further along, but it is in general hard to read. To facilitate evaluation of the results median, 15th, and 85th percentile, maximum, and minimum values were used. In Figure F.14 results are presented from calculating sensitivity index related to the complete inclination channel. Referring to equation F.2, all inputs in the inclination channel are varied by $\Delta S = 0.1$, which is 10 percent of the total input scale, as the inputs are scaled between 0 and 1. This is akin to using a miscalibrated sensor where an offset is introduced to all inputs on a channel.

Particularly interesting insight is that while sensitivity of the inclination input starts slightly below 1 with very little change in median value, other channels' sensitivity index starts at zero for the first predicted data point (one example in Figure F.15) and the magnitude of it rises as the prediction is done for further points. Both of those results indicate that the developed model acts as it should. The output is a continuous prediction, a continuation of the inclination input, therefore necessarily if whole input channel is shifted by a given value, the first output will shift by nearly identical value. Inclination cannot change sharply, only gradually, hence near zero sensitivity on the other three channels for the first prediction point.

For channels other than inclination, sensitivity index grows as the prediction distance increases. This is consistent with the modelled process, as the inclination can be influenced more in the outputs farther from the initiated gap.

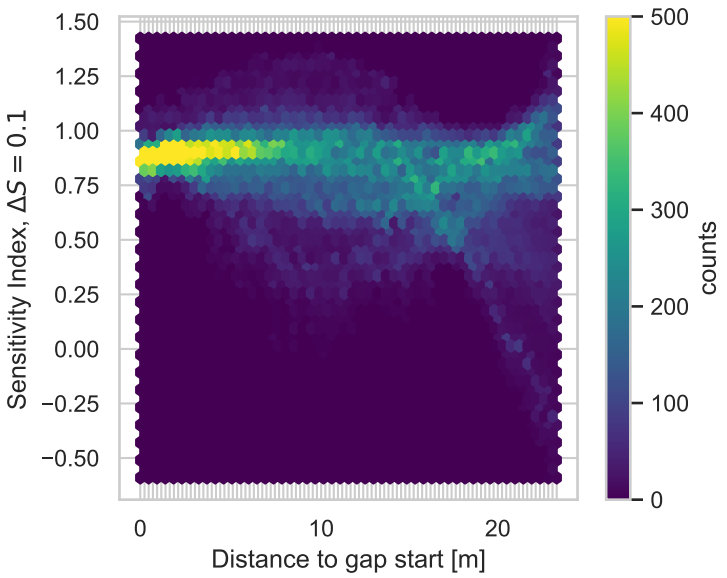


Figure F.13: Statistical sensitivity to 10 percent value change on Inclination channel, heatmap.

F.3.3 Magnitude of change

In calculating sensitivity index there is a potential influence of the ΔS value. To evaluate this matter, sensitivity index was calculated at $\Delta S = 0.1$ and $\Delta S = 0.5$. It must be noted that this value corresponds to scaled parameters, hence are equivalent to 10 percent and 50 percent of full scale change respectively. Results can be inspected in Figure F.16. Median sensitivity can be considered the same for both selected ΔS. The main difference is that the result is more stable, i.e. the 15th and 85th percentile values are closer to median. This suggests that the selection of specific ΔS does not affect the results of the analysis significantly.

F.3.4 Sensitivity to single inputs

Another type of sensitivity analysis proposed is sensitivity to single inputs, which can also uncover sensitivity to outliers or sensitivity to a specific input. A function

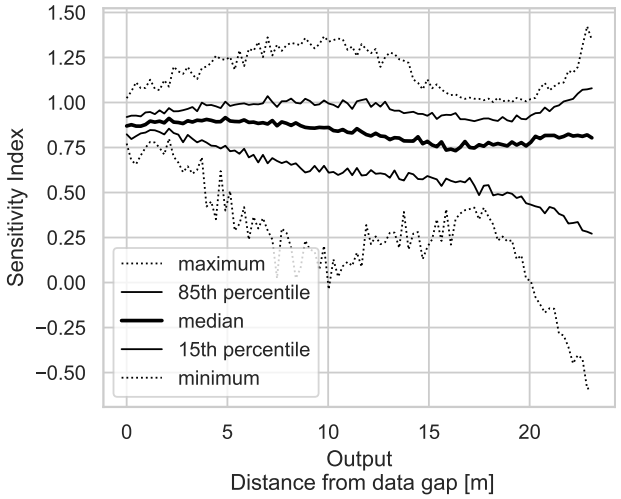


Figure F.14: Statistical sensitivity to 10 percent value change on Inclination channel, distribution.

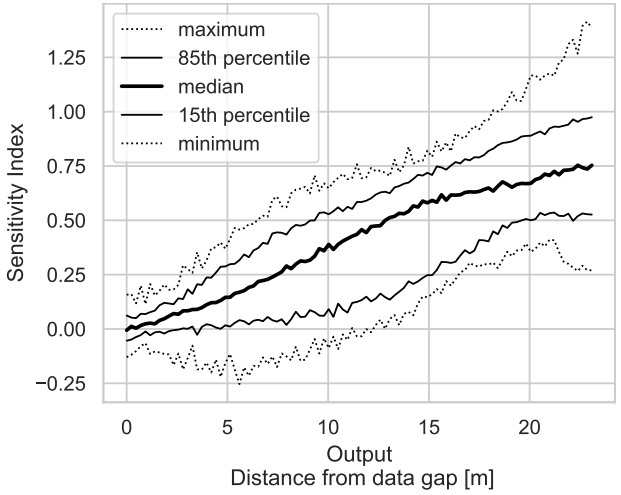


Figure F.15: Statistical sensitivity to 10 percent value change on Torque channel.

was developed that calculated the sensitivity index, as per equation F.2, again using the existing dataset of inputs.

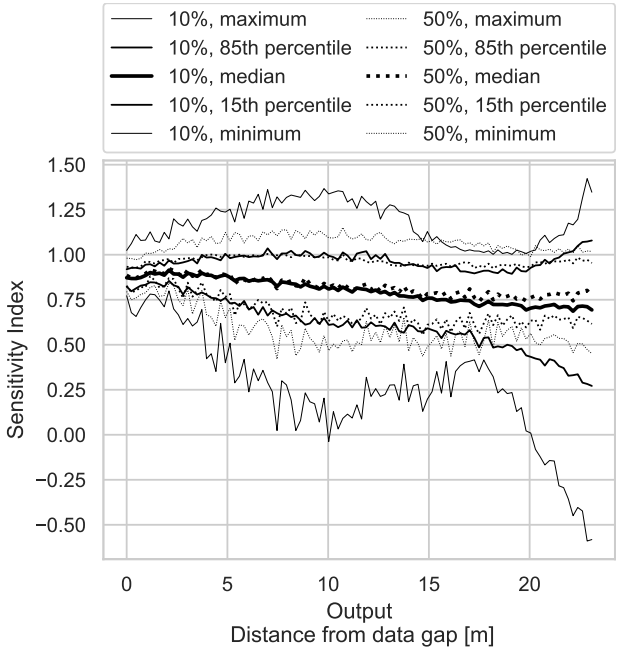


Figure F.16: Evaluation of the importance of magnitude of change in sensitivity analysis, inclination channel.

$$SI_t^A(k) = \frac{f(X_t^A + \Delta S_k, X_t^B, X_t^C, X_t^D) - f(X_t^A - \Delta S_k, X_t^B, X_t^C, X_t^D)}{2\Delta s} \quad \text{(F.13)}$$

where

$$\Delta S_k = \begin{bmatrix} \overset{1}{0} & \overset{2}{0} & \overset{...}{...} & \overset{k-1}{0} & \overset{k}{1} & \overset{k+1}{0} & \overset{...}{...} & \overset{643}{0} \end{bmatrix}^T \quad \text{(F.14)}$$

Per given sample only one singular input was modified. This was done for all singular inputs and channels separately, from the first input to the last one, altering original value by *0.1*.

To provide a two dimensional, easy to read chart, for this method sensitivity that is averaged over all outputs is used. This way, an indication of how all the outputs change on average in relation to a change on a specific input is provided.

For further clarity result plots are bundled into input channels. Two charts are reproduced here, one for the inclination channel, Figure F.17, and one for the torque channel, Figure F.19. What is interesting in the first figure, is that there is a decrease in value close to the input points at about -3 meters from the gap. Additionally, points farther than -10 meters have much lower response, which suggests that it may be possible to shorter the input to the model without a significant loss in performance. It must be noted, though, that the case study model was developed through hyperparameter tuning and the input length of 23 meters was selected as providing the smallest error. Figure F.17 suggests that this window can be shortened with minimal influence on the output, and therefore minimal influence on the error. Such changes may be considered if, for example, shortening that input would generate significant savings in the deployment phase of a project. It may also be used in guiding sensor accuracy selection, by showing relationship between measurement error and output. Additionally, charts similar to Figure F.17 and Figure F.19 may be generated while tuning hyperparameters of a data-driven model, giving an instant insight in the inner workings of the neural network, allowing to narrow down a range of possible hyperparameters.

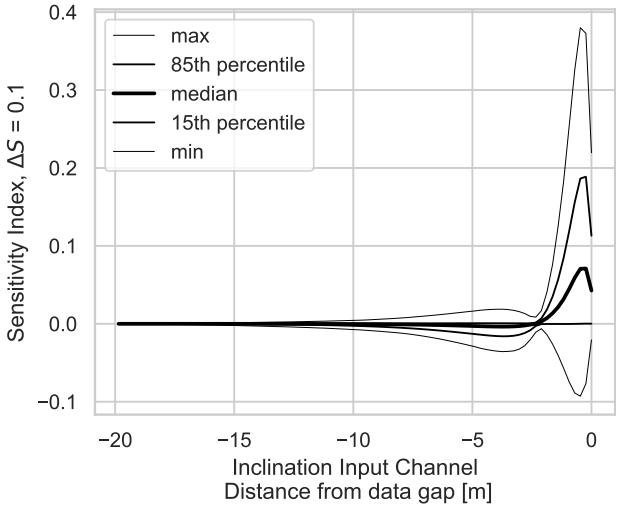


Figure F.17: Output channel sensitivity to a change in a singular point on inclination input channel.

To explore results from Figure F.17 further, a chart was generated that investigates the point of maximum sensitivity at approximately -1 meters from data gap, presented in Figure F.18. This allows us to evaluate how the point of maximum sensitivity, in relation to all outputs, is distributed between individual outputs. One can see that output at approximately 8 meters after the data gap is the most sensitive, with further points' sensitivity dropping. This is in line with previously explored sensitivity of the inclination channel as a whole in relation to individual outputs, Figure F.14. Additional takeaway here is that 15th percentile line lies almost perfectly at zero sensitivity, suggesting that there is a relatively sharp cutoff with sporadic negative sensitivity, which is in line with the physical aspect of the model, i.e. higher inclination just before the data gap will suggest higher inclination after it.

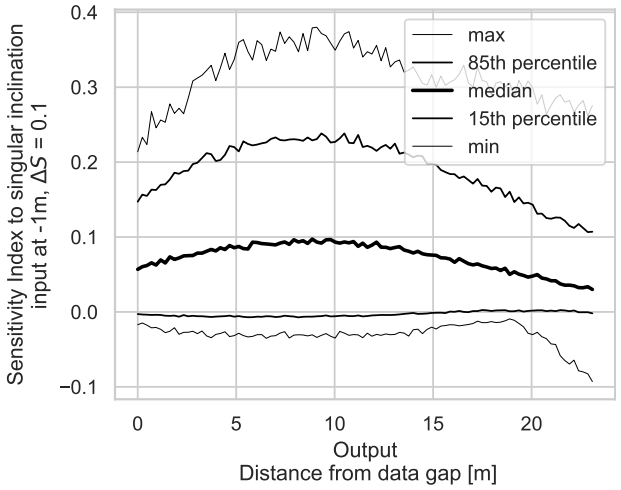


Figure F.18: Sensitivity to inclination input at -1 m to data gap, point of maximum sensitivity

Similar analysis was performed for input channel related to torque, with results in Figure F.19. Note that this channel has more inputs, as in our case study this parameter is considered available both before and after the data gap, while the inclination only before. Interestingly, there is a clear minimum between -6 meters and 3 meters. This is approximately the same area that has a local maximum in the inclination input. Higher response is visible for the points from 0 meters

forward. This again confirms the intuitive behaviour of the model. Physically, the inputs above 0 meters are co-located with the output, and therefore logically will have the highest impact.

Another interesting insight here is that again the 15th percentile line often lies at zero change, also in line with the physics of the modelled system. Attention must be paid to the scale of the y-axis on these charts, as the sensitivity to the inclination input channel is an order of magnitude higher than the one calculated for *rotary speed*.

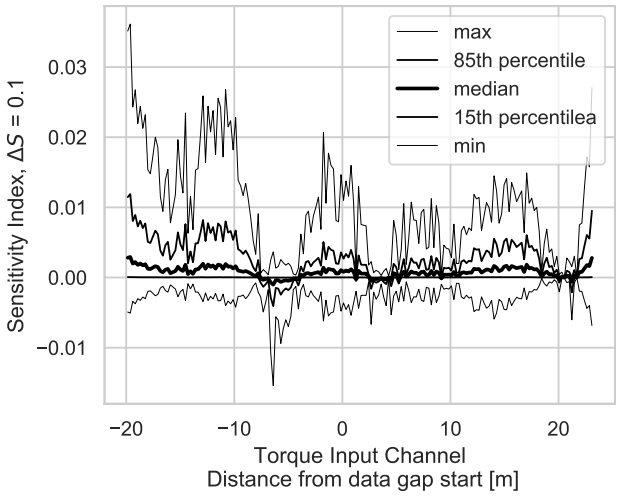


Figure F.19: Output channel sensitivity to a change in a singular point on torque input channel.

F.3.5 Input-output heatmap

It is possible to plot median sensitivity of outputs to specific input channel as a heatmap. This allows to see the influence of individual inputs to individual outputs. It must be highlighted that this is done on a point-by-point basis, one-at-a-time. Note that in this case the color scales are not synchronized between the charts as the results vary by an order of magnitude. Due to nature of the plot it is the median sensitivity calculated over the complete dataset that is visualized.

Looking at Figure F.20 one can consider it an expanded version of Figure F.17, with the output dimension now visible. One can, for example, identify, that inclination input at -5 meters from the gap positively influences the output close to the gap, but negatively influences the outputs further from the gap. Similarly, in Figure F.21, the weakest response between inputs between -6 meters and -3 meters can be identified, first seen in Figure F.19, as a white plateau over the whole range of outputs. In this figure one can also identify a potential problem with the analyzed model. As explained before, this input physically overlaps with the output. Therefore input at location of n meters cannot have influence on the outputs before location n meters. There practically should be a triangle between points (0,0), (23,0) and (23,23) with zero sensitivity - yellow overlay in that area was added in Figure F.21.

This apparent contradiction can be explained through correlation and causation. It is likely that during operation it is the achieved inclination (here *output*) that influences the *rotary speed* (here *input*), and not the other way round. The correlation however still exists and the model can use it. This means that the model will correctly predict inclination during typical drilling, but will fail if the inputs will deviate too much from normal operation. This is understandable considering that the training dataset covers only typical drilling.

F.3.6 Channel sensitivity as a function of well depth

The core of the presented method is statistical analysis of the models' sensitivity with the training/validation/testing dataset at its foundation. It is, however, possible to investigate how the calculated statistical sensitivity is affected by the contents of available dataset. Our case study models BHA-formation interaction while drilling a curved section of a well. Sensitivity analysis can be performed on a channel as a whole, as done in the section *Sensitivity Index of Input Channels*, and evaluated along the length of the well. Results are presented in Figure F.22. The sensitivity analysis was performed for a complete channel, as previously presented in Figures F.14 and F.15 and using Equation F.11. Sensitivity was

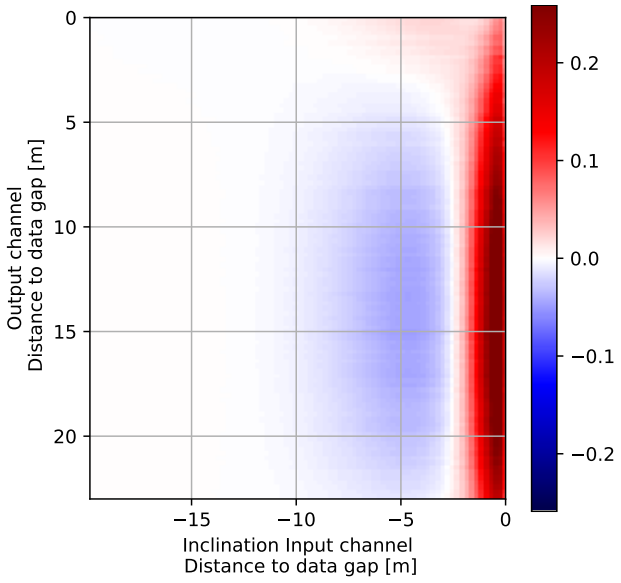


Figure F.20: Sensitivity heatmap between Inclination input channel and output.

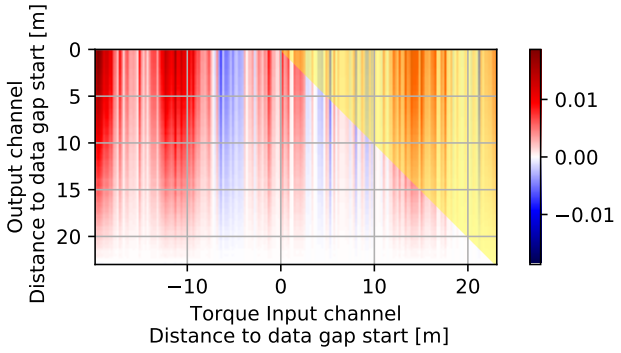


Figure F.21: Sensitivity heatmap between Torque input channel and output. Yellow overlay shows area that should have sensitivity zero.

calculated separately for all four channels. The heatmap presented in Figure F.22 shows the median value of sensitivity for a given output.

Such a graphic enables evaluation of sensitivity changes with the varying base value in the dataset. It is possible to identify areas of significantly high or low sensitivity and investigate those specific samples. This can provide further insights of how the system behaves - in our case study it is possible separately

analyze the sensitivity during rotating and sliding modes of the BHA. It is also possible to evaluate how well the dataset represents potential inputs of the system by looking for unusual responses, with an assumption that unusual sensitivity response coincides with unusual set of inputs. In our case this can be seen in the inclination channel between samples 500 and 600 and further on at approximately sample 950, where there is negative inclination channel sensitivity in outputs close to 23 meters. This is the area responsible for minimum sensitivity below zero visible earlier in Figure F.14.

F.3.7 Comparison to Sobol' indices.

Sobol' indices developed in the early 90s [1] are a form of variance-based global sensitivity analysis. This method is used to calculate how much of the output's variance can be attributed to different inputs. With this method one can calculate how important are individual inputs, and if second order indices are calculated, how important is interaction between two inputs.

Our data-driven sensitivity analysis method used *n=1300* samples. To calculate sensitivity index 2n model evaluations must be performed, as a base value has to be increased and decreased. With 643 inputs, to calculate sensitivity for all of them individually, this results in *1 674 400* evaluations. For comparison, calculated Sobol' indices were calculated using SALib Sensitivity Analysis Library in Python [26], using methods developed by Sobol and Saltelli [27, 28, 19]. *n=5000* samples were used, limited by memory in available computer. Note that in common practice 10^4 samples are required to estimate Sobol' indices within 10% uncertainty [4]. Our setup resulted in *3 225 000* sub-samples for first order calculations. It took 10 minutes to generate required samples, 38 minutes to evaluate the model for all the samples and 40 minutes to calculate the resulting indices[3]. Second order calculations require double the amount of samples and were not calculated. Other, faster algorithms were considered, however in general

[3]Calculations were performed on Intel Core i7-8850H, 32GB RAM, Python 3.6.9, SALib library in version 1.3.8.

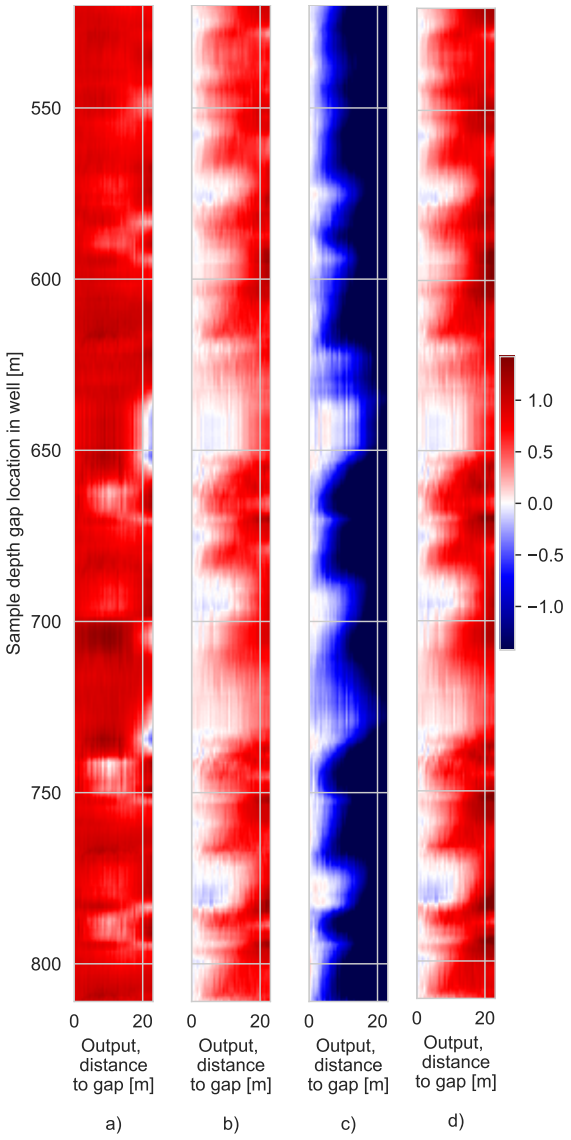


Figure F.22: Channel sensitivity per sample, for: a) Sensitivity over Inclination channel, b) Surface torque channel, c) Rotary speed channel, d) Rate of Penetration channel

literature suggests that those methods remain costly, unstable and biased for models with more than 10 inputs [29, 4].

While computation time was not significantly different between our method and

Sobol' indices, our method allows for evaluating partial results, that is sensitivity of a single input can be calculated for a model with thousands of inputs very quickly. Note that since our case study model has 100 outputs representing 100 individual depth-steps in the model, each input will have 100 Sobol indices related to all separate inputs. Sobol indices are calculated assuming singular output; since our case study produces 100 outputs as a complete depth-sequence prediction, Sobol indices had to be calculated as if there were 100 separate models with single output to evaluate. Minimum and maximum boundaries for Sobol' indices calculation were taken from boundary values for specific input found in the dataset. This is typically in range *(0,1)* for all inputs except for the ones responsible for inclination input. Due to scaling and moving the center of coordinate system employed in the model they vary in minimums and maximums.

For comparison purposes, a number of figures that were originally developed with our proposed method were recreated to compare results. Note that it is not possible to evaluate the sensitivity of a complete channel (multi-input) with Sobol' indices, hence it is not possible to compare those. First comparison is between our Figure F.17 and Figure F.23 generated with Sobol' index data. It can be seen that results are different. First, Sobol indices in principle cannot be negative, hence negative sensitivity at -5 meters is not seen. The chart overall looks much more noisy and difficult to read.

Chart type that was used to investigate the maximum sensitivity on that channel, Figure F.18, was also plotted again, which can be seen in Figure F.24. Note that there is no statistical distribution to the new chart, as at this level singular Sobol' indices are being plotted.

As with previous comparison, general insights are still here, but some of the information available with our data-driven method is not there. Our method uncovers the spread in sensitivity, is based on actual data, and additionally uncovers that in some rare cases the sensitivity can be negative, which is not visible using Sobol' indices.

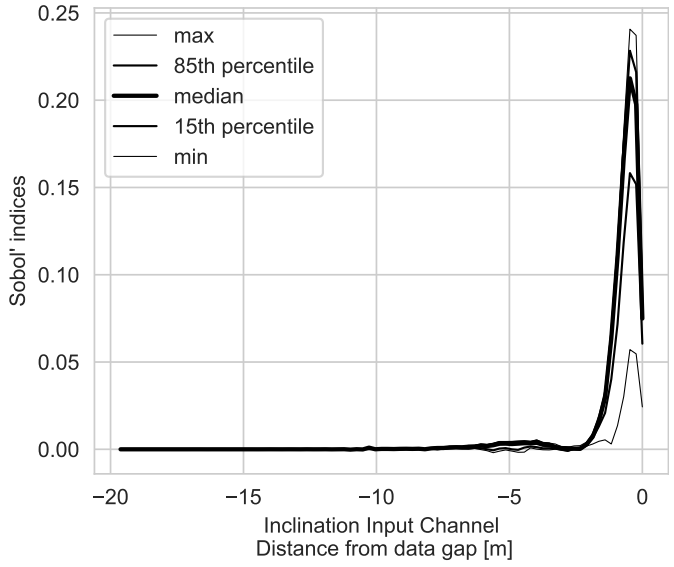


Figure F.23: Average sobol indices for inclination inputs.

Lastly, the Sensitivity heatmap for inclination channel, Figure F.20, was re-done as a heatmap of Sobol' indices in Figure F.25. When using our method one can see exactly which areas are sensitive, and which are not. Using Sobol' indices this information is again missing. While our method uncovered an area of negative sensitivity, it is nearly invisible in Figure F.25, and the information about the sign of sensitivity is necessarily gone.

F.4 Discussion

Approaching sensitivity analysis in the domain of neural networks, especially those with a high number of inputs, require a novel approach. This is especially visible in the case of recurrent neural networks that have highly interdependent inputs; it is where variance-based methods cannot be reliably used [30]. Proposed data-driven approach exploits the fact that ML methods are created when there is a dataset available that is sufficiently big to cover the possible inputs and outputs.

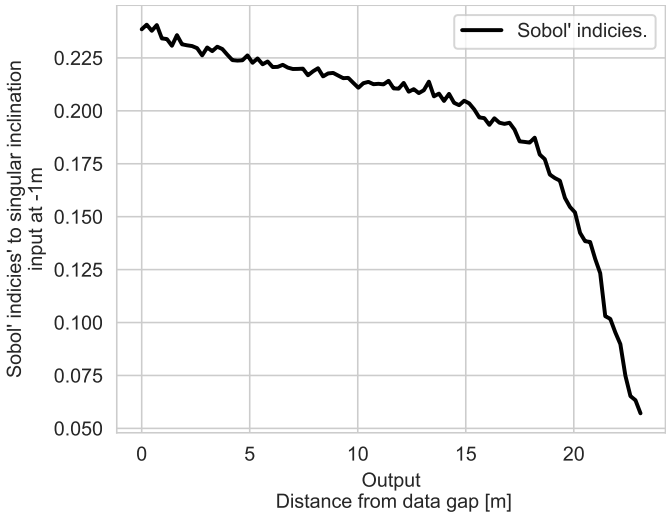


Figure F.24: Sobol indices for all individual outputs for inclination input at -1m from data gap.

While geometric arguments are brought up proving insufficiency of one at a time methods [19] due to inability to cover the possible inputs, the proposed method does not suffer from this shortcoming. By performing analysis on a single input, as in the section on *Sensitivity to single inputs*, through repeating calculations on multiple starting points, the whole scope of potential inputs is covered. It is crucial to understand that the presented method deals with sensitivity of the model, not sensitivity of the real system. It is a significant distinction and expert interpretation is needed to decide whether a specific insight belongs to the system, or does it only exist in the model.

Additional benefit of data-driven sensitivity analysis is mathematical simplicity of the approach. It utilizes very basic methods and expands them through repetition. It becomes more and more common to bypass math-heavy approaches where analytical solutions exist with Monte Carlo type methods due to their simplicity [31]. Additionally, since the basic output of the presented method is a dataset, as opposed to a single value, further, deeper analysis is possible than presented here. For example, referring back to Figure F.14, median sensitivity of all outputs is the same, approximately 0.8. It is however possible to extract much more

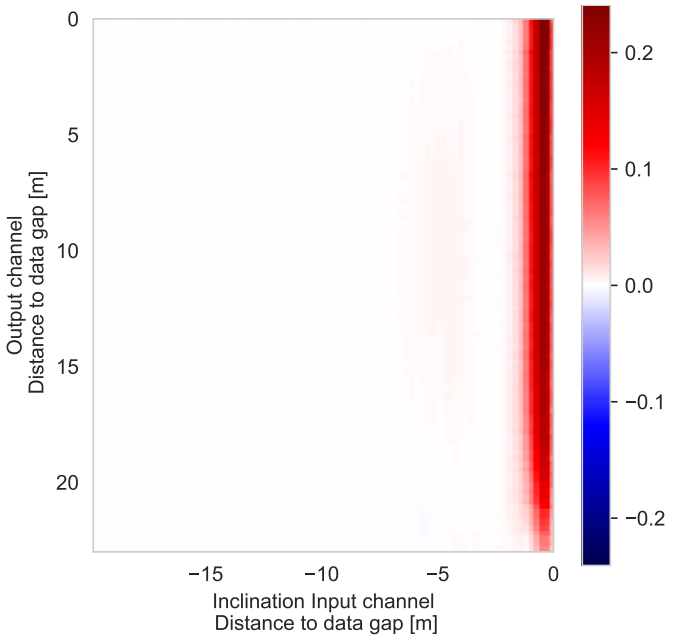


Figure F.25: Sobol indices heatmap of Inclination input channel and output.

information, such as standard deviation or variance of the sensitivity. Result of such calculation is shown in Figure F.26, where standard deviation is shown for sensitivity on the inclination channel. This gives a very good indication of how different sensitivity can be between outputs, even though the median value is the same. Such calculations can be tailor made for each case, providing even more tools to analyse a model compared to common methods.

F.5 Conclusion

Classical sensitivity analysis methods are often ill-suited for data-driven models, such as recurrent neural networks, due to high number of interdependent inputs. Common, basic, one-at-a-time methods fail to provide meaningful results when high number of inputs is considered. These can, however, be enhanced by employing datasets originally used for training, validation, and testing the

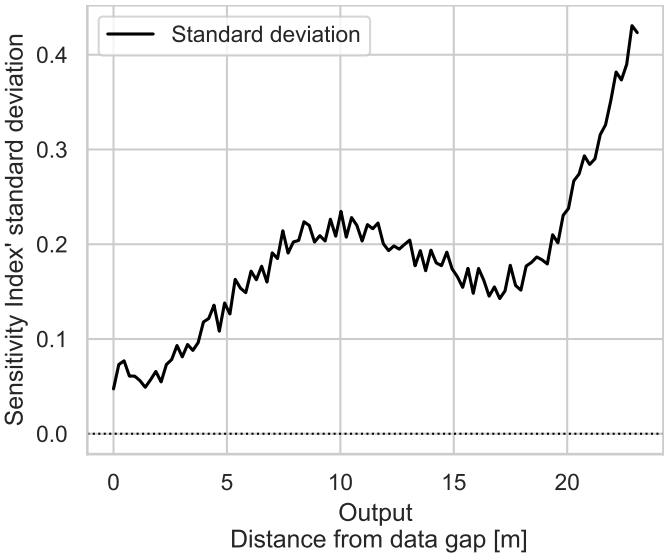


Figure F.26: Statistical sensitivity, Inclination channel, standard deviation

models. These datasets necessarily cover the possible inputs, often with statistical distribution matching reality.

In our case study, the proposed sensitivity analysis method was able to pinpoint many new insights into performance and limitations of the model that would otherwise be difficult to uncover. It was discovered that sensitivity exists where it should not. This suggests that the model performs well when directional driller follows a certain procedure of operation, and the model can exploit correlations that are tied to that behaviour. At the same time, the model would perform poorly if used as a simulator, where users are free to do as they please. We believe that the presented method may be a very useful tool in developing ever more complicated machine learning models, as well as, in evaluation of other black box solutions.

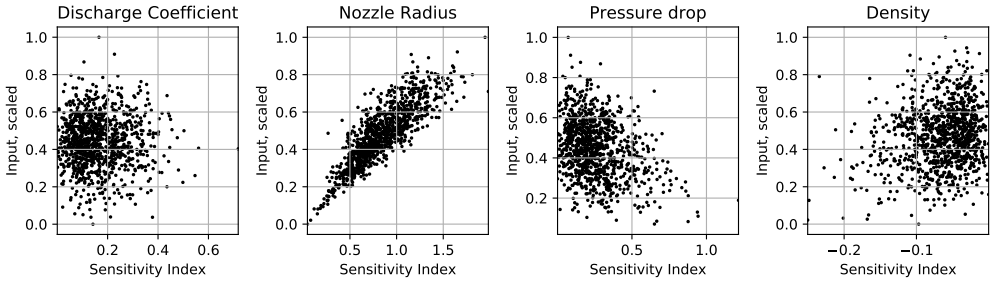


Figure F.27: Sensitivity index, scaled function, scatterplot.

Appendix

Elasticity Index as input exponent

Deriviation of Elasticity Index being the exponent of the input:

Consider Elasticity index of a function $y = f(x)$ as:

$$\mathcal{E}_{y,x} = \frac{dy}{dx} \cdot \frac{x}{y} = f'(x) \cdot \frac{x}{y} \tag{F.15}$$

A function in a general form $f(x) = C \cdot x^{\alpha}$ has derivative in relation to x as $f'(x) = \alpha \cdot C \cdot x^{\alpha-1}$. Therefore $\mathcal{E}_{y,x} = \alpha \cdot C \cdot x^{\alpha-1} \cdot \frac{x}{C \cdot x^{\alpha}}$. Since $x^{\alpha-1} = \frac{x^{\alpha}}{x}$:

$$\mathcal{E}_{y,x} = \alpha \cdot C \cdot \frac{x^{\alpha}}{x} \cdot \frac{x}{C \cdot x^{\alpha}} \tag{F.16}$$

which can be reduced to simply $E_{y,x} = \alpha$. Bear in mind, that this is true for ΔS being significantly smaller than S_0.

Additional figures

Some less important figures are reproduced in the appendix only.

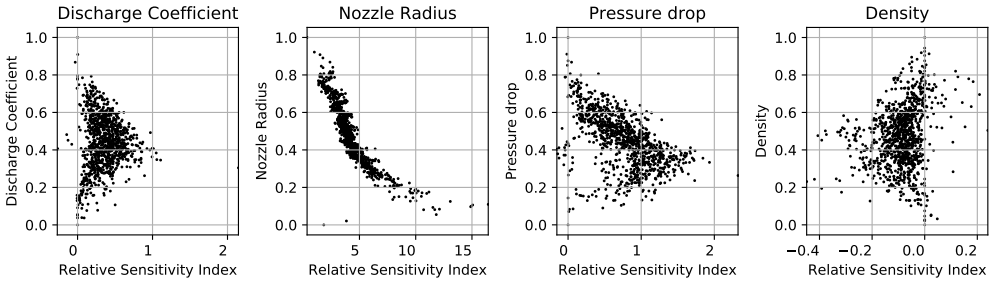


Figure F.28: ML model, relative Sensitivity Index.

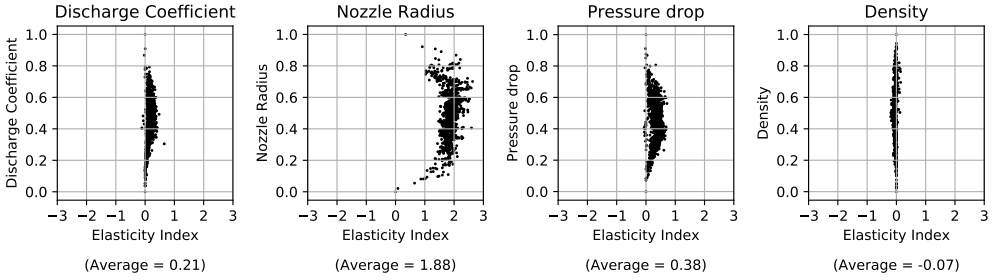


Figure F.29: ML model, Elasticity Index

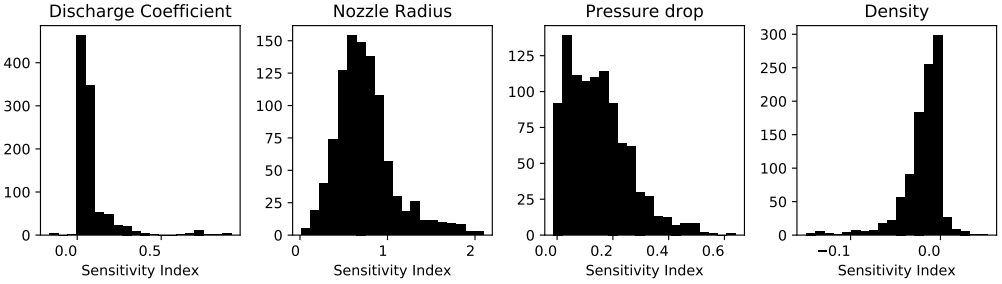


Figure F.30: Uneven sample distribution, sensitivity index histogram

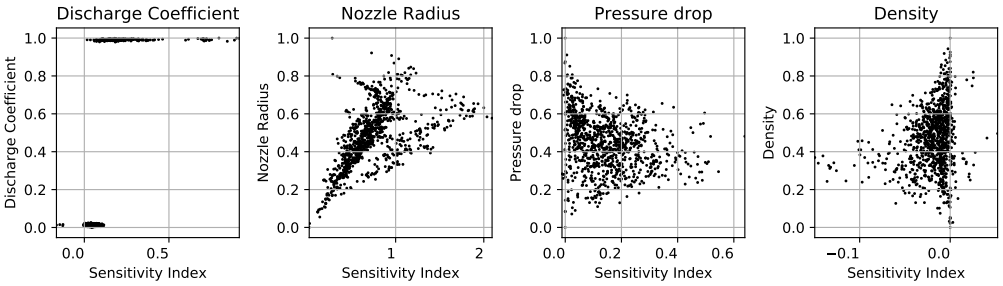


Figure F.31: Uneven sample distribution, sensitivity index scatterplot

Algorithm F.2: Data-Driven Sensitivity Index, per channel

Data: $X := [X_1, X_2, \ldots, X_n]$
$X_n := [X_{n,1}, X_{n,2}, \ldots, X_{n,k}]$
$X_{n,k} := [X_{n,k,1}, X_{n,k,2}, \ldots, X_{n,k,m_k}]$
an array of n *samples with* k *channels, and* m_k *inputs on k-th channel.*
Δ typically *0.1, or 10% of input range*

Result: $DDSI := [SI_1, SI_2, \ldots, SI_n]$
$SI_n := [SI_{n,1}, SI_{n,2}, \ldots, SI_{n,k}]$
$SI_{n,k} := [SI_{n,k,1}, SI_{n,k,2}, \ldots, SI_{n,k,w}]$
an array of individual, local Sensitivity Indices through n *samples,* k *input channels, and* w *outputs.*

```
1  Function Model(
     [[x_{n_i},1,1, x_{n_i},1,2, ..., x_{n_i},1,m_k],
     [x_{n_i},2,1, x_{n_i},2,2, ..., x_{n_i},2,m_k,],
     ...,
     [x_{n_i},k,1, x_{n_i},k,2, ..., x_{n_i},k,m_k]]):
2      model calculations;
3      return [y1,y2, ... ,yw] ;
4  for i:=1 to n do
5      for r:=1 to k do
6          samplePlus = X[i];
7          sampleMinus = X[i];
8          for j:=1 to m_k do
9              samplePlus[r][j] += Δ;
10             sampleMinus[r][j] -= Δ;
11         outputPlus = model(samplePlus);
12         outputMinus = model(sampleMinus);
13         oneSensitivity = (outputPlus - outputMinus) / 2Δ ;
14         DDSI[i,r] = oneSensitivity;
```

Bibliography

[1] Ilya M Sobol. "Sensitivity Estimates for Nonlinear Mathematical Models". In: *Mathematical modelling and computational experiments* 1.4 (1993), pp. 407–414.

[2] Dorota Kurowicka and Roger Cooke. "Uncertainty Analysis with High Dimensional Dependence Modelling". In: *Uncertainty Analysis with High Dimensional Dependence Modelling. Wiley Series in Probability and Statistics*. Jan. 2006. ISBN: 0-470-86306-4. DOI: 10.1002/0470863072.

[3] T. W. Simpson, J. D. Peplinski, P. N. Koch, and J. K. Allen. "Metamodels for Computer-Based Engineering Design: Survey and Recommendations". In: *Engineering with Computers* 17.2 (2001), pp. 129–150. ISSN: 01770667. DOI: 10.1007/PL00007198.

[4] Bertrand Iooss and Paul Lemaître. "A Review on Global Sensitivity Analysis Methods". In: *Operations Research/ Computer Science Interfaces Series* 59 (2015), pp. 101–122. ISSN: 1387666X. DOI: 10.1007/978-1-4899-7547-8\{_\}5.

[5] Majdi I. Radaideh, Stuti Surani, Daniel O'Grady, and Tomasz Kozlowski. "Shapley Effect Application for Variance-Based Sensitivity Analysis of the Few-Group Cross-Sections". In: *Annals of Nuclear Energy* 129 (2019), pp. 264–279. ISSN: 18732100. DOI: 10.1016/j.anucene.2019.02.002. URL: https://doi.org/10.1016/j.anucene.2019.02.002.

[6] Alsakran Jamal, Rodan Ali, Alhindawi Nouh, and Faris Hossam. "Visualization Analysis of Feed Forward Neural Network Input Contribution". In: *Scientific Research and Essays* 9.14 (July 2014), pp. 645–651. ISSN: 1992-2248. DOI: 10.5897/SRE2014.5895. URL: http://academicjournals.org/journal/SRE/article-abstract/F21242846447.

[7] Daniel P. Loucks, Eelco van Beek, Jery R. Stedinger, Jozef P.M. Dijkman, and Monique T. Villars. "9 Model Sensitivity and Uncertainty Analysis". In: *Water Resources Systems Planning and Management: An Introduction to Methods, Models and Applications*. 2005, pp. 254–290. ISBN: 92-3-103998-9. DOI: ISBN:92-3-103998-9. pmid: 25246403. URL: https://www.utwente.nl/ctw/wem/education/afstuderen/Loucks_VanBeek/09_chapter09.pdf.

[8] Erman Ulker and Mehmet Sorgun. "Comparison of Computational Intelligence Models for Cuttings Transport in Horizontal and Deviated Wells". In: *Journal of Petroleum Science and Engineering* 146 (Oct. 2016), pp. 832–837. ISSN: 09204105. DOI: 10.1016/j.petrol.2016.07.022.

[9] Mehmet Sorgun and Erman Ulker. "Modeling and Experimental Study of Solid-Liquid Two-Phase Pressure Drop in Horizontal Wellbores with Pipe Rotation". In: *Journal of Energy Resources Technology, Transactions of the ASME* 138.2 (Mar. 2016). ISSN: 15288994. DOI: 10.1115/1.4031743.

[10] Russel E. Caflisch. "Monte Carlo and Quasi-Monte Carlo Methods". In: *Acta Numerica* 7 (1998), pp. 1–49. ISSN: 14740508. DOI: 10.1017/S0962492900002804.

[11] Andrzej T Tunkiel, Tomasz Wiktorski, and Dan Sui. "Continuous Drilling Sensor Data Reconstruction and Prediction via Recurrent Neural Networks". In: *ASME 2020 39th International Conference on Ocean, Offshore and Arctic Engineering*. 2020. DOI: 10.1115/OMAE2020-18154.

[12] Ming Lu, S. M. AbouRizk, and U. H. Hermann. "Sensitivity Analysis of Neural Networks in Spool Fabrication Productivity Studies". In: *Journal of Computing in Civil Engineering* 15.4 (Oct. 2001), pp. 299–308.

ISSN: 0887-3801. DOI: 10.1061/(ASCE)0887-3801(2001)15:4(299). URL: http://ascelibrary.org/doi/10.1061/%28ASCE%290887-3801%282001%2915%3A4%28299%29.

[13] Kristen Bell Detienne, David H Detienne, and Shirish A Joshi. "Neural Networks as Statistical Tools for Business Researchers". In: *Organizational Research Methods* 6.2 (2003), pp. 236–265. DOI: 10.1177/1094428103251907. URL: https://doi.org/10.1177/1094428103251907.

[14] Vahid Nourani and Mina Sayyah Fard. "Sensitivity Analysis of the Artificial Neural Network Outputs in Simulation of the Evaporation Process at Different Climatologic Regimes". In: *Advances in Engineering Software* (2012). ISSN: 09659978. DOI: 10.1016/j.advengsoft.2011.12.014.

[15] Peyman Babakhani, Jonathan Bridge, Ruey-an Doong, and Tanapon Phenrat. "Parameterization and Prediction of Nanoparticle Transport in Porous Media: A Reanalysis Using Artificial Neural Network". In: *Water Resources Research* 53.6 (2017), pp. 4564–4585. DOI: 10.1002/2016WR020358. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2016WR020358.

[16] Sarit Dutta and J. P. Gupta. "PVT Correlations for Indian Crude Using Artificial Neural Networks". In: *Journal of Petroleum Science and Engineering* 72.1-2 (2010), pp. 93–109. ISSN: 09204105. DOI: 10.1016/j.petrol.2010.03.007. URL: http://dx.doi.org/10.1016/j.petrol.2010.03.007.

[17] Mohammad Hasan Shojaeefard, Mostafa Akbari, Mojtaba Tahani, and Foad Farhani. "Sensitivity Analysis of the Artificial Neural Network Outputs in Friction Stir Lap Joining of Aluminum to Brass". In: *Advances in Materials Science and Engineering* 2013 (2013). Ed. by Rui Vilar, p. 574914. ISSN: 1687-8434. DOI: 10.1155/2013/574914. URL: https://doi.org/10.1155/2013/574914.

[18] Simone Franceschini, Lorenzo Tancioni, Massimo Lorenzoni, Francesco Mattei, and Michele Scardi. "An Ecologically Constrained Procedure for Sensitivity Analysis of Artificial Neural Networks and Other Empirical Models". In: *PloS one* 14.1 (Jan. 2019), e0211445–e0211445. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0211445. URL: https://pubmed.ncbi.nlm.nih.gov/30699204%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6353184/.

[19] Andrea Saltelli and Paola Annoni. "How to Avoid a Perfunctory Sensitivity Analysis". In: *Environmental Modelling and Software* 25.12 (2010), pp. 1508–1517. ISSN: 13648152. DOI: 10.1016/j.envsoft.2010.04.012. URL: http://dx.doi.org/10.1016/j.envsoft.2010.04.012.

[20] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures". In: *Neural Computation* 31.7 (2019), pp. 1235–1270. DOI: 10.1162/neco\{_\}a\{_\}01199. URL: https://doi.org/10.1162/neco_a_01199.

[21] Knut Sydsaeter and Peter J Hammond. *Mathematics for Economic Analysis.* HB135 S98. 1995.

[22] Leo Breiman. "Random Forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[23] Andrzej T Tunkiel. *Github for OMAE2020-18154.* URL: https://github.com/AndrzejTunkiel/OMAE2020-18154.

[24] Equinor. *Volve Field Data (CC BY-NC-SA 4.0).* 2018. URL: https://www.equinor.com/en/news/14jun2018-disclosing-volve-data.html.

[25] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation". In: *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference* (2014), pp. 1724–1734. DOI: 10.3115/v1/d14-1179.

[26] Jon Herman and Will Usher. "SALib: An Open-Source Python Library for Sensitivity Analysis". In: *The Journal of Open Source Software* 2.9 (Jan. 2017). DOI: 10.21105/joss.00097. URL: https://doi.org/10.21105/joss.00097.

[27] I. M. Sobol. "Global Sensitivity Indices for Nonlinear Mathematical Models and Their Monte Carlo Estimates". In: *Mathematics and Computers in Simulation* 55.1-3 (2001), pp. 271–280. ISSN: 03784754. DOI: 10.1016/S0378-4754(00)00270-6.

[28] Andrea Saltelli. "Making Best Use of Model Evaluations to Compute Sensitivity Indices". In: *Computer Physics Communications* (2002). ISSN: 00104655. DOI: 10.1016/S0010-4655(02)00280-1.

[29] Jean Yves Tissot and Clémentine Prieur. "Bias Correction for the Estimation of Sensitivity Indices Based on Random Balance Designs". In: *Reliability Engineering and System Safety* 107 (2012), pp. 205–213. ISSN: 09518320. DOI: 10.1016/j.ress.2012.06.010.

[30] Andrea Saltelli and Stefano Tarantola. "On the Relative Importance of Input Factors in Mathematical Models". In: *Journal of the American Statistical Association* 97.459 (2002), pp. 702–709. DOI: 10.1198/016214502388618447. URL: https://doi.org/10.1198/016214502388618447.

[31] Jake VanderPlas. *Statistics for Hackers*. 2016. URL: https://www.youtube.com/watch?v=Iq9DzN6mvYA.