Journal of Petroleum Science and Engineering 208 (2022) 109760

Contents lists available at ScienceDirect

Journal of Petroleum Science and Engineering

journal homepage: www.elsevier.com/locate/petrol

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

Andrzej T. Tunkiel [a,*], Dan Sui [a], Tomasz Wiktorski [b]

[a] Department of Energy and Petroleum Engineering, Faculty of Science and Technology, University of Stavanger, 4036 Stavanger, Postboks, 8600 Forus, Norway
[b] Department of Electrical Engineering and Computer Science, Faculty of Science and Technology, University of Stavanger, 4036 Stavanger, Postboks, 8600 Forus, Norway

ARTICLE INFO

*Keywords:*
Recurrent neural network
Machine learning
Data pre-processing methods
Continuous learning
Real-time logs
Drilling

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.

## 1. Introduction

### 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 (Barbosa et al., 2019) 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 (Equinor, 2018), expose that they can be occupied in over 80% by data gaps (Tunkiel et al., 2020d). Drilling logs from this dataset are used as a case study throughout this paper.

To capture the temporal behavior of a given logged attribute recurrent neural networks (RNN) are commonly used (Rumelhart et al., 1986). Basic architecture of such network is shown in Fig. 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 Fig. 1. Various designs of a RNN cell exist, with Long Short-Term Memory (LSTM) (Yu et al., 2019) and Gated Recurrent Units (GRU) (Chung et al., 2014) seeing wide implementations.

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

* Corresponding author.
*E-mail address:* andrzej.t.tunkiel@uis.no (A.T. Tunkiel).
*URL:* http://www.ux.uis.no/~atunkiel/ (A.T. Tunkiel).

https://doi.org/10.1016/j.petrol.2021.109760
Received 2 July 2021; Received in revised form 1 October 2021; Accepted 29 October 2021
Available online 13 November 2021

**Nomenclature**

| | |
|---|---|
| $\delta_{cf}$ | Cutoff value |
| $\Delta X$ | Dataset, first numerical derivative |
| $\dot{x}$ | 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) |
| $SS_g$ | 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 $g(\cdot)$ |
| $MSE_g$ | 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 neighbor |
| GC | Gap coefficient |
| GRU | Gated recurrent unit |
| K | Quantity of neighbors in KNN |
| k | Quantity of datapoints of the first numerical derivative dataset |
| KNN | K-nearest neighbors |
| 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 Neighbor 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 |

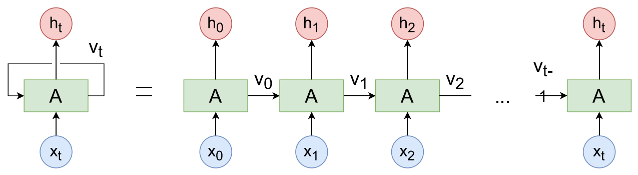


**Fig. 1.** Recurrent Neural Network basic architecture.

this step is often simply mentioned without even indicating which algorithm or the parameters are used (Song et al., 2020). To the best of authors' knowledge, petroleum related papers on the RNN rarely mention resampling (Chhantyal et al., 2018; Osarogiagbon et al., 2020); our previous work on the RNN (Tunkiel et al., 2021) 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" (Lipton et al., 2015) 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 modeling. 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.

## *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 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, modeling 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 (Tunkiel et al., 2021, 2020c), made target attribute agnostic, and the whole framework is open sourced and published on Github (Tunkiel, 000). 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)

### 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).

## 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 1 as well as presented in a flowchart form in Fig. 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 behavior 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 (Tunkiel et al., 2020a). 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.

**Table 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 |

### 2.1. Data selection

#### 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 (Equinor, 2018) 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 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, i.e. 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

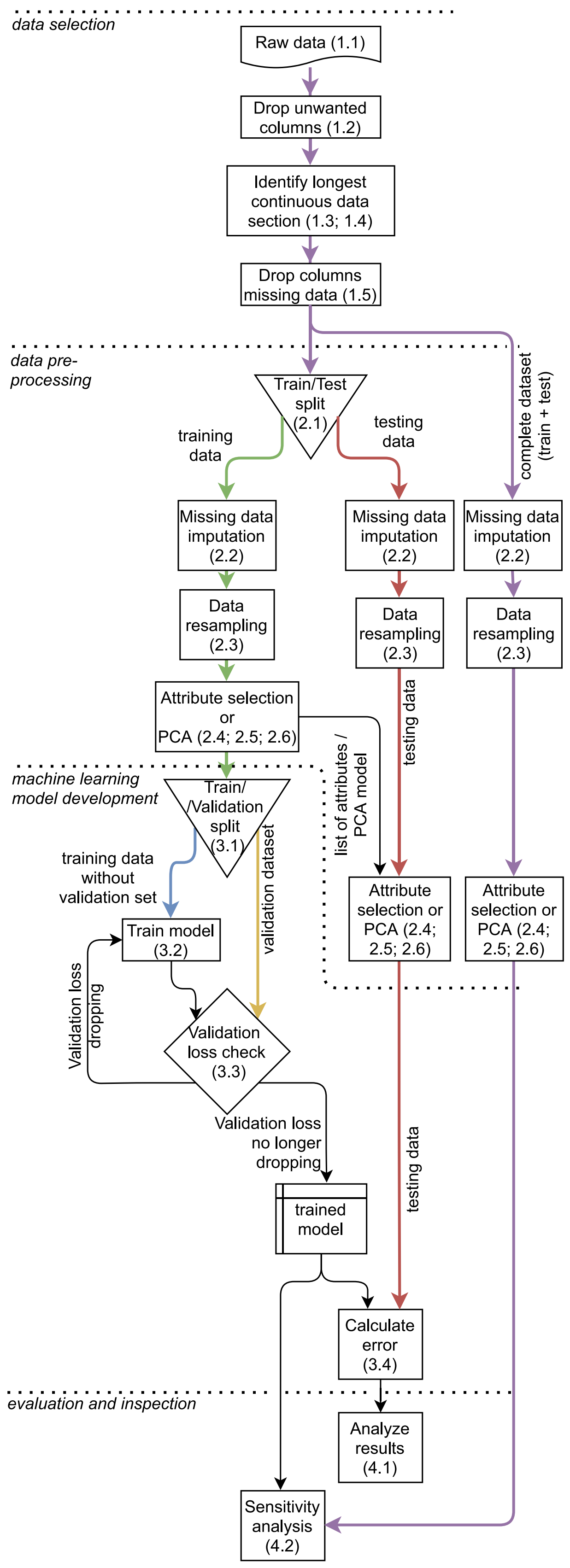

data selection
Raw data (1.1)
Drop unwanted columns (1.2)
Identify longest continuous data section (1.3; 1.4)
Drop columns missing data (1.5)
data pre-processing
Train/Test split (2.1)
training data
testing data
complete dataset (train + test)
Missing data imputation (2.2)
Data resampling (2.3)
Attribute selection or PCA (2.4; 2.5; 2.6)
list of attributes / PCA model
testing data
machine learning model development
Train/Validation split (3.1)
training data without validation set
validation dataset
Train model (3.2)
Validation loss dropping
Validation loss check (3.3)
Validation loss no longer dropping
trained model
Attribute selection or PCA (2.4; 2.5; 2.6)
Calculate error (3.4)
evaluation and inspection
Analyze results (4.1)
Sensitivity analysis (4.2)

**Fig. 2.** Process framework flowchart, color coded datasets.

initial filtering and outlier 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 (Sui et al., 2018; Geekiyanage et al., 2018, 2020) are worth considering.

### 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

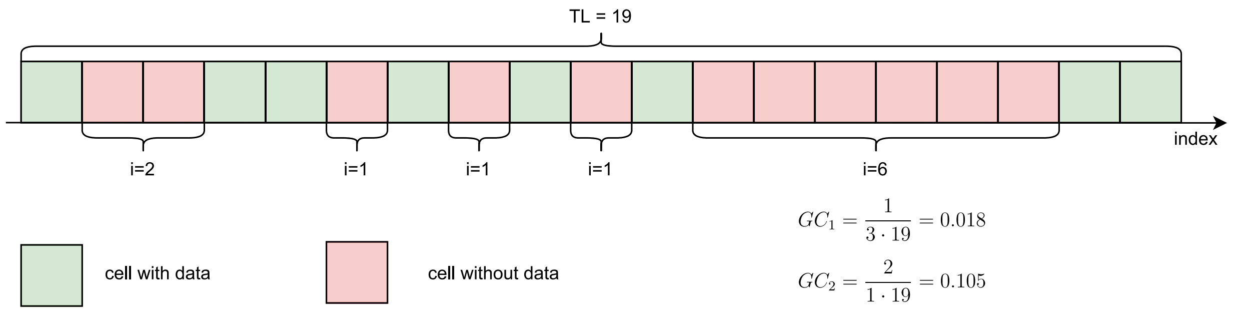
The azimuth data was removed as it is delayed due to the same reasons as inclination data that is to be predicted. Raw sensor data also exists in the dataset for both inclination and azimuth, which has to be removed, as the target is a simple function of those, also delayed, readings. Additionally index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

### 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 analyzing 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:

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



$$GC_1 = \frac{1}{3 \cdot 19} = 0.018$$

$$GC_2 = \frac{2}{1 \cdot 19} = 0.105$$

$$GC_6 = \frac{6}{1 \cdot 19} = 0.316$$

**Fig. 3.** Gap coefficient calculation example.

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 Fig. 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:

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

All gap sizes for all attributes are calculated separately. A cutoff value $\delta_{cf}$ 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 plentiful gaps will yield a low GC value, while few big gaps will return a high GC. In our case studies, $\delta_{cf}$ is set as 0.005, but for other datasets different value may be more appropriate.

### 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.

### 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.

## 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.

### 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 (Tunkiel et al., 2020b), 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.

### 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.

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

[1] Base case value, evaluated further in the manuscript.

*leak* future information (Kaufman et al., 2011). 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.

#### 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.

#### 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 (Geller, 2019) to scaling, normalizing, or standardizing 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) (Pearson, 1901) 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 (Pedregosa et al., 2011) 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 (Stacklies et al., 2007) 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.

#### 2.2.5. Select attributes, PCA

Two methods of selecting attributes are implemented and evaluated: Pearson coefficient (Benesty et al., 2009), ppscore (Wetschoreck et al., 2020) which provides asymmetric correlation of coefficients based on decision trees. Additionally, Principal Component Analysis (Pearson, 1901), 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.

#### 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

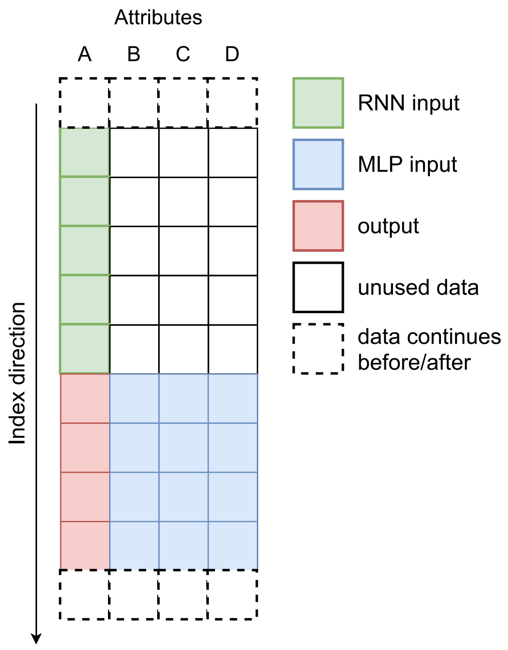


**Fig. 4.** Shape of the data showing RNN and Multilayer Perceptron (MLP) input shapes.

in the prediction area are also selected as input of a single sample. For better visualization of the sample shape Fig. 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 **n** + **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 **m** rows in the prediction area and **j** attributes, which are all selected attributes from the dataset (or PCA dimensions). The exact quantity of **m** and **n** rows will depend on the length of the prediction and the memory area (defined by user), as well as re-sampling rate. In Fig. 4, $m = 4$, $n = 5$ and $j = 3$.

### 2.3. Applying ML model

#### 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 (Pasini, 2015). 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 (Pasini, 2015). 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 (Esmaeilzadeh et al., 2020, 2019).

#### 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 Fig. 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 0.25 m 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) (Rosenblatt, 1961) artificial network, while left hand side branch consists of Recurrent Neural Network employing Gated Recurrent Units (Chung et al., 2014) (GRU). While in the presented case study and evaluation further in the manuscript GRU was selected 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 (Hochreiter and Schmidhuber, 1997). 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 Fig. 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.

#### 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 (Prechelt, 1998) and is widely used in machine learning. A specified number of epochs is set as *patience*, which defines the amount of epochs since the last validation loss improvement is allowed. In presented implementation patience of 50 epochs was used.

#### 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 model, for multiple points in the dataset simulating continuous drilling of a well, i.e. *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.

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

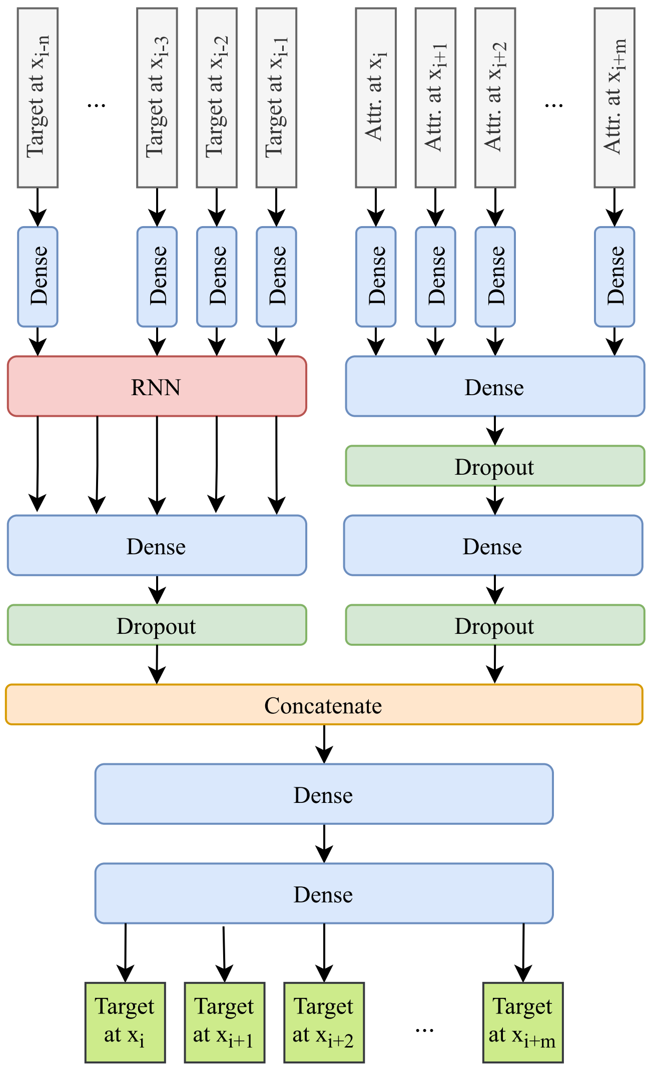


**Fig. 5.** Simplified model schematic.

### 2.4. Model evaluation and inspection

#### 2.4.1. Results analysis

Results are evaluated by incorporating training-while-drilling methodology (Tunkiel et al., 2021) derived from a continuous learning concept (Liu, 2017). 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. Fig. 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 Fig. 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 0 m 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

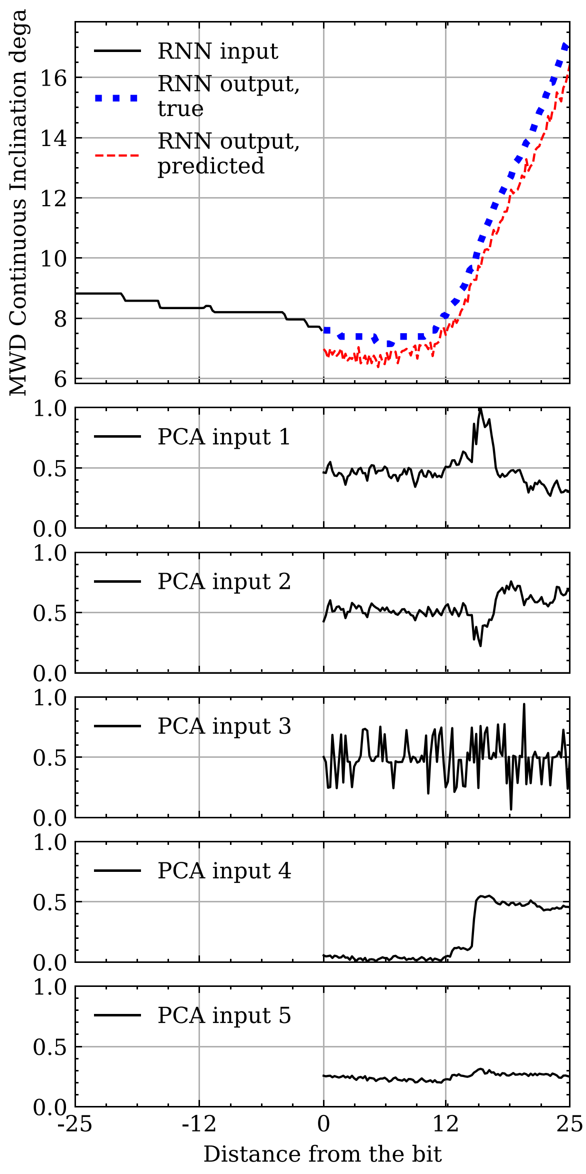


**Fig. 6.** Example of inputs, outputs, and predicted values. Inclination prediction case study. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

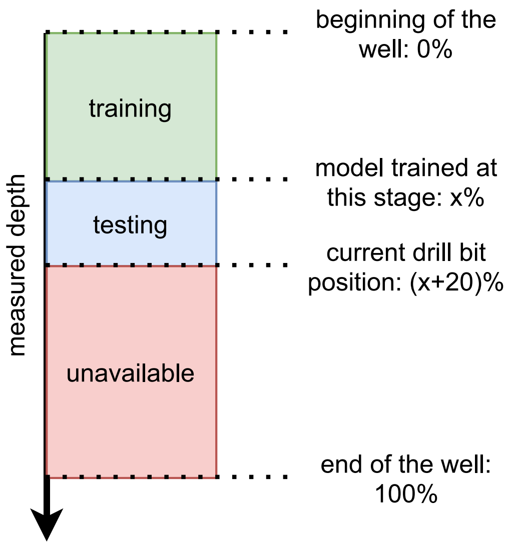

beginning of the well: 0%
training
model trained at this stage: x%
testing
current drill bit position: (x+20)%
measured depth
unavailable
end of the well: 100%

**Fig. 7.** Training-while-drilling, train-test data split example.

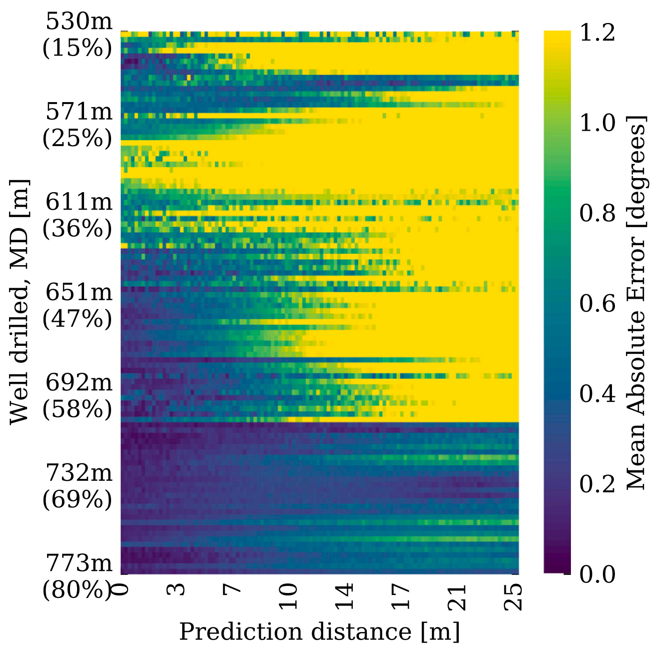


**Fig. 8.** Example results of continuous learning applied to predicting 25 meters ahead of the bit.

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 (Tunkiel et al., 2020c), where, referring to Fig. 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, 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 (Tunkiel et al., 2021).

### 2.4.2. *Sensitivity analysis*

Sensitivity analysis of the model can be performed using Data-driven Sensitivity Analysis methodology (Tunkiel et al., 2020a), 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 statistically. More traditional methods, such as sobol indices (Sobol, 1993) 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.

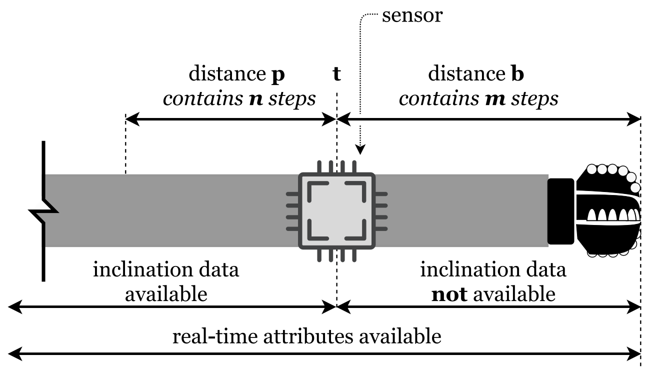

sensor
distance **p**
*contains **n** steps*
**t**
distance **b**
*contains **m** steps*
inclination data available
inclination data **not** available
real-time attributes available

**Fig. 9.** Sensor lag case study.

### 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.

## 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 Eq. (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 Fig. 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 0.05 m.

Proposed GC data is presented in Fig. 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 2.1.4. 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.

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 (Landau, 1967).

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 $\Delta X$. 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

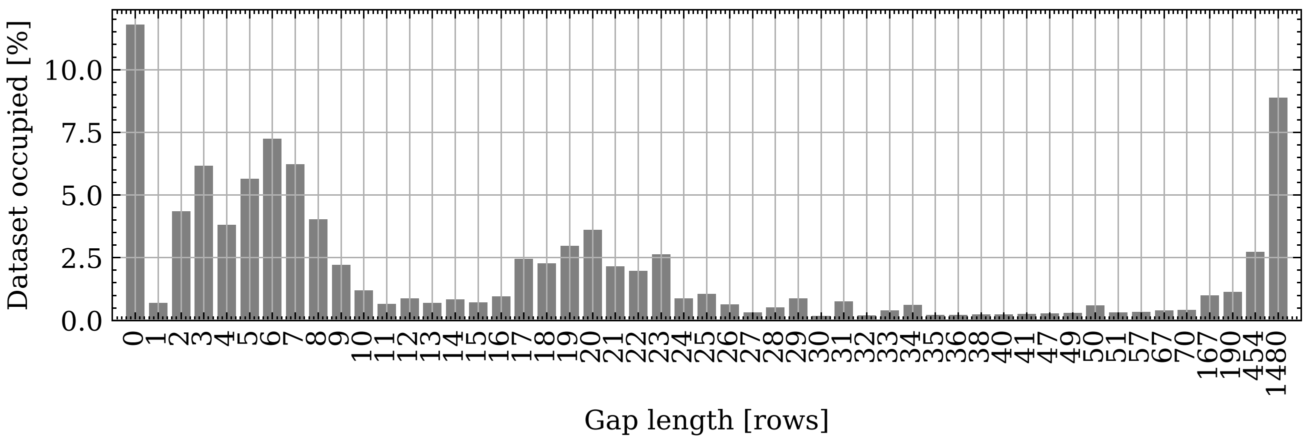


**Fig. 10.** Raw gap statistics, MWD Continuous Inclination, Volve well F9a.

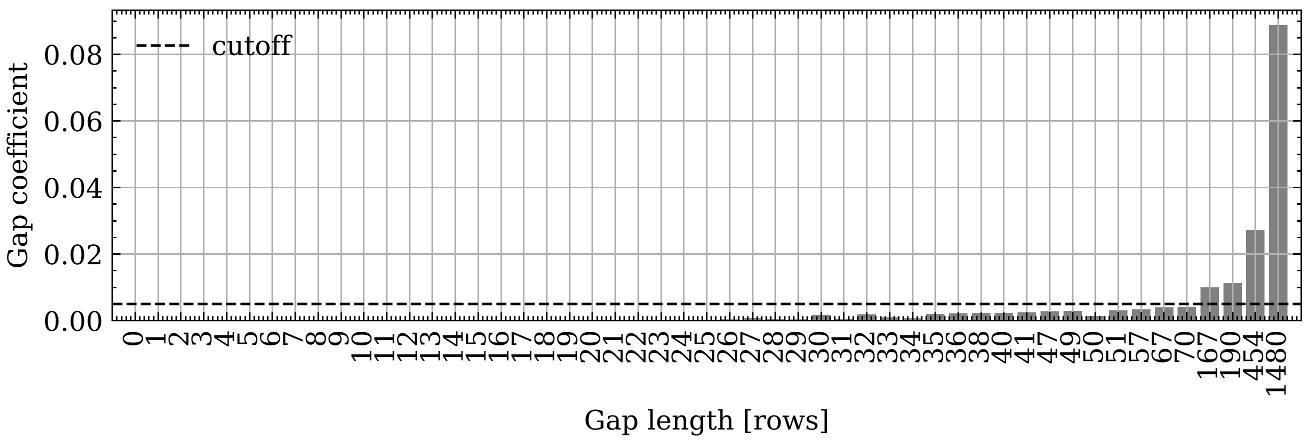


**Fig. 11.** Gap statistics, gap coefficient, MWD Continuous Inclination, Volve well F9a.

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 totaling 12 650 train-test iterations, the mean absolute error was analyzed, with results presented in Fig. 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 Fig. 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 Fig. 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.

## 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 (Neil et al., 2016) that mitigates the issue, the asynchronous nature of recorded value remains a practical problem. Additionally, the RNN architectures implementing

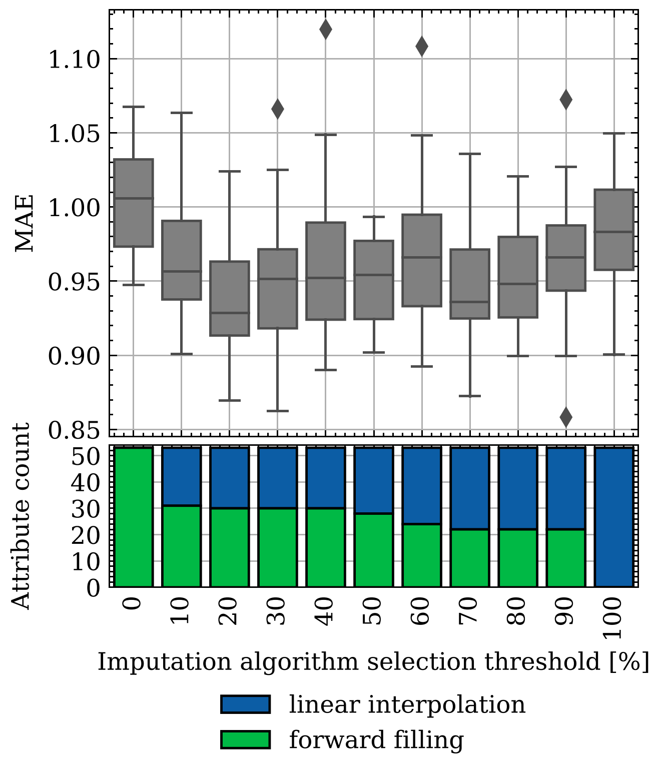


**Fig. 12.** Smart filling study, inclination prediction, by varying the threshold for applying linear interpolation versus forward filling, the percentage of zero-valued first numerical derivative values. (0% - Forward Fill only, 100% - linear interpolation only).

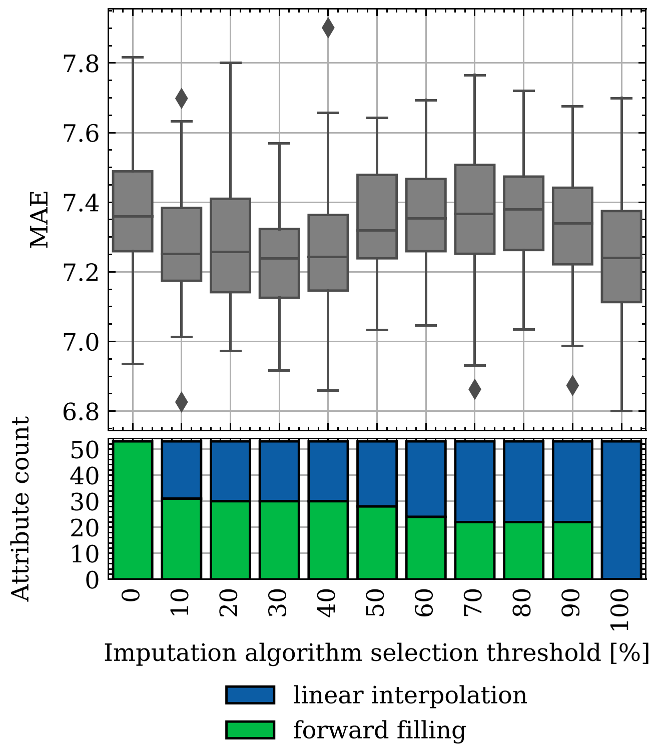


**Fig. 13.** Smart filling study, ROP prediction, by varying the threshold for applying linear interpolation versus forward filling, the percentage of zero-valued first numerical derivative values. (0% - Forward Fill only, 100% - linear interpolation only).

such time gate are not included, as for now, in the most common machine learning libraries such as Tensorflow/Keras or PyTorch.

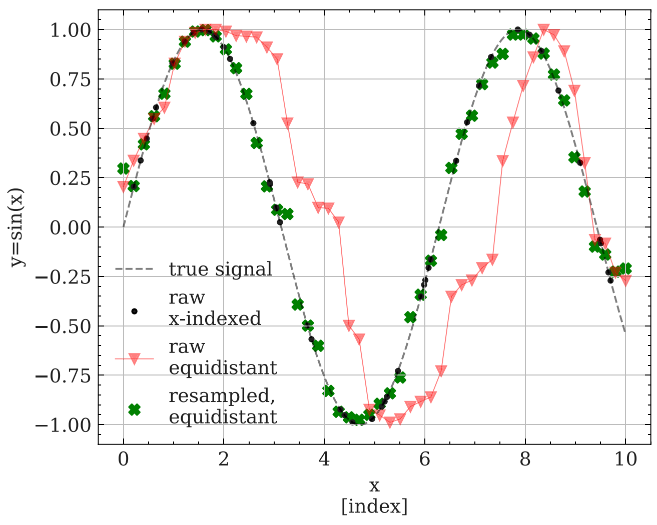


**Fig. 14.** Importance of resampling, example of signal deformation due to uneven sampling rate.

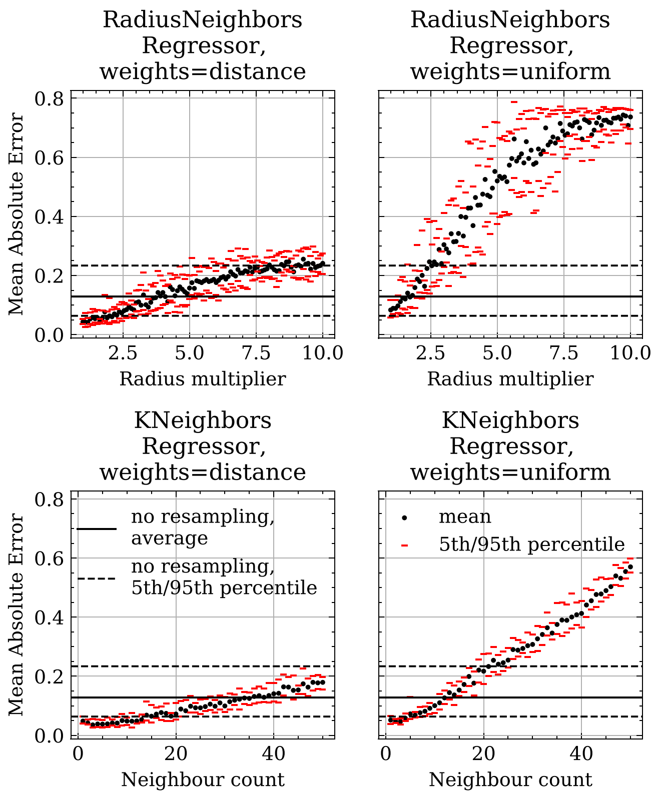


**Fig. 15.** Resampling error study.

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.

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

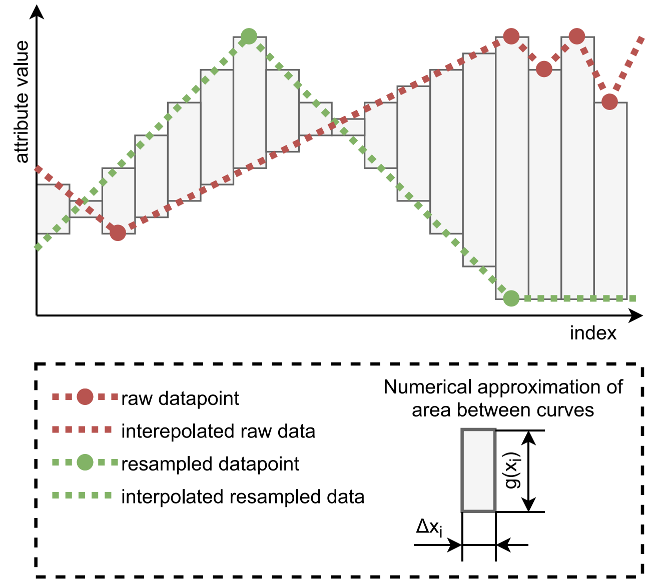


**Fig. 16.** Numerical difference visualization.

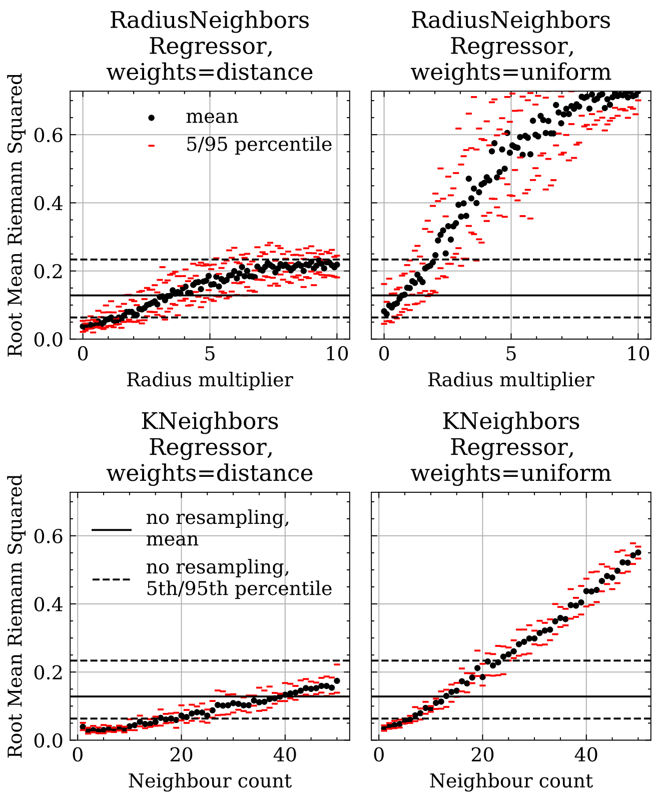


**Fig. 17.** Resampling error study.

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.

### 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 Fig. 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 Fig. 14 such data is calculated using K-Nearest Neighbor (KNN) regression (Fix, 1985) algorithm for neighbor 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 (Fix, 1985), and Fixed Radius Neighbor (FRN) (Bentley, 1975). 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-learn implementation of these algorithms was used (Pedregosa et al., 2011).

### 4.2. Resampling quality evaluation

#### 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 Fig. 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 Neighbors Regressor implementation and the FRN using Radius Neighbors Regressor, from scikit-learn implementation (Pedregosa et al., 2011); 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 neighboring data along $x$-axis.

Since the test was based on randomly generated values along the $x$-axis, a Monte Carlo simulation (Caflisch, 1998) 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

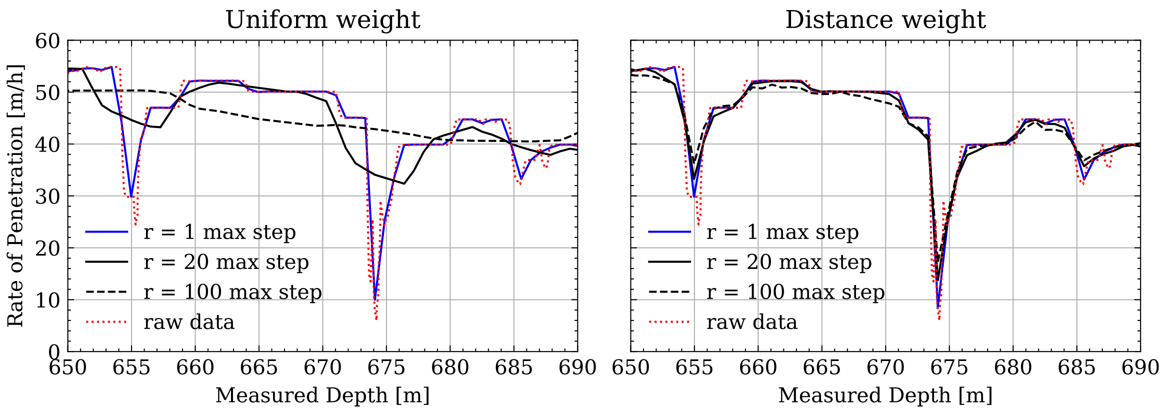


**Fig. 18.** Radius Neighbor Regressor, effects of different regressor radius on resampled data.

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 Fig. 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.

### 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 us 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

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

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 Fig. 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 do not match. To solve this issue the squared difference between the two functions is calculated discretely.

The practical implementation is based on Riemann sum (Engelke and Sealey, 2009). A basic example of the method for calculating the discrete difference between the curves is shown in Fig. 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:

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

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

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

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):

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

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

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

**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 Fig. 15. Error calculated using RMRS matches, for all intents and purposes, the calculations using the ground truth nearly perfectly, as seen in Fig. 17 which is nearly identical to Fig. 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 Figs. A.26, A.27, and A.28 with more modest results of $R^2$ between 0.945 and 0.997.

## 4.3. *Resampling: Case study*

### 4.3.1. *Resampling quality evaluation on drilling data*

To visualize the difference between the raw and resampled data in practice Figs. 18 and 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-Nearest Neighbor algorithms were applied at three

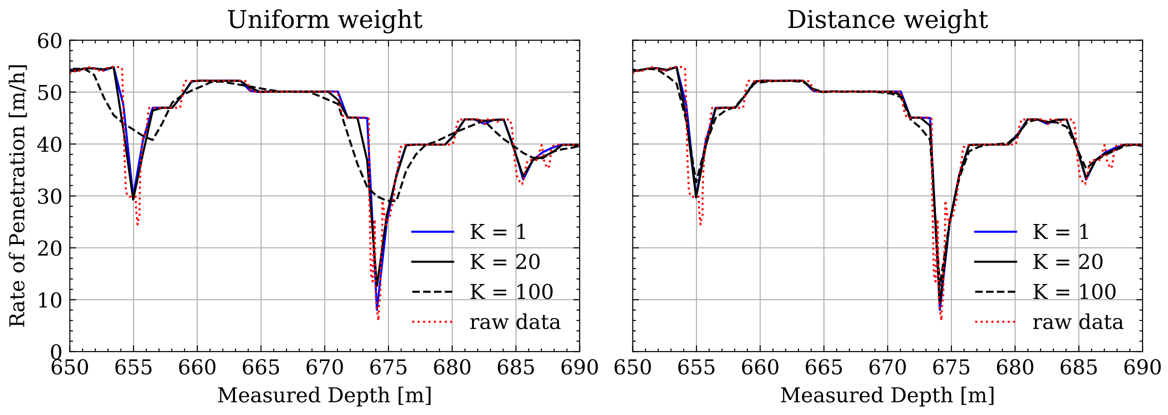


**Fig. 19.** K-Nearest Neighbors, effects of different neighbor count on resampled data.

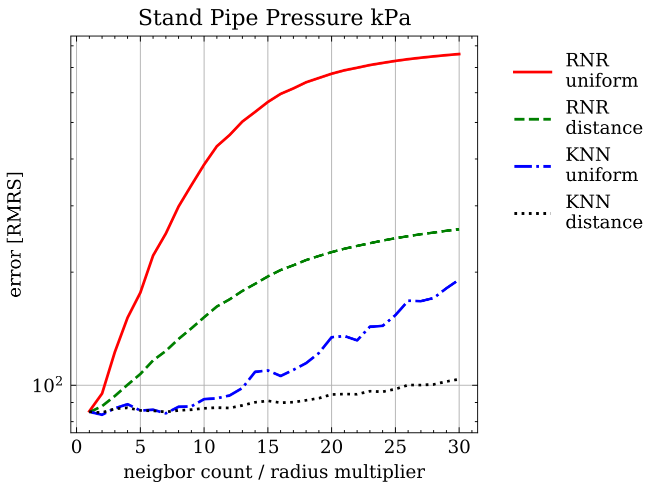


**Fig. 20.** Evaluation using RMRS example.

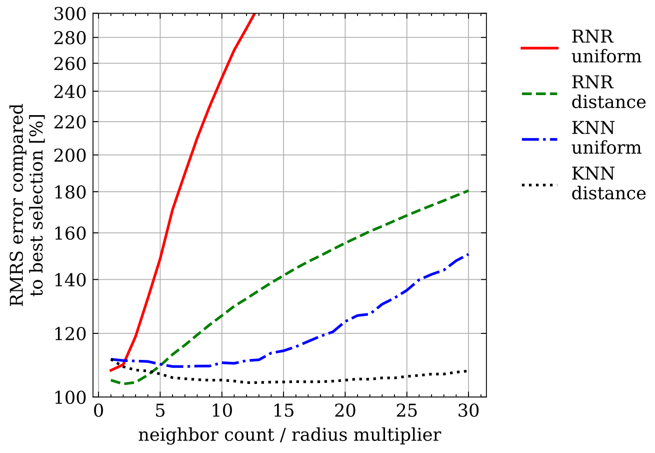


**Fig. 21.** Comparing a fixed algorithm throughout the dataset versus always picking best one.

different configurations of maximum depth-step multipliers of 1, 20, and 100 as well as of neighbor count of 1, 20 and 100. When the uniform weight is used then even the relatively small radius distorts the signal.

In Fig. 18, inspecting signal for resampling using distance-based weights it 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 behavior 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 Eq. (7) is shown in Fig. 20. Here, a Stand Pipe Pressure (SPP) attribute from the well F9a of the Volve dataset is evaluated for four different resampling algorithms, Radius Neighbor Regressor and K-Nearest Neighbor with both uniform and distance based weights. Different radiuses and neighbor 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 neighbor 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.

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 Table 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 Fig. 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 Fig. 21. These results show that if one resampling algorithm was to be used for all algorithms "*blindly*" it is best to either use Radius Neighbor Regressor, distance weighted, with radius of 2 times the biggest original index step, or K-Nearest Neighbor, distance weighted, with 15 neighbors. Either of those two options come with penalty of under 5 percent of additional error compared to best

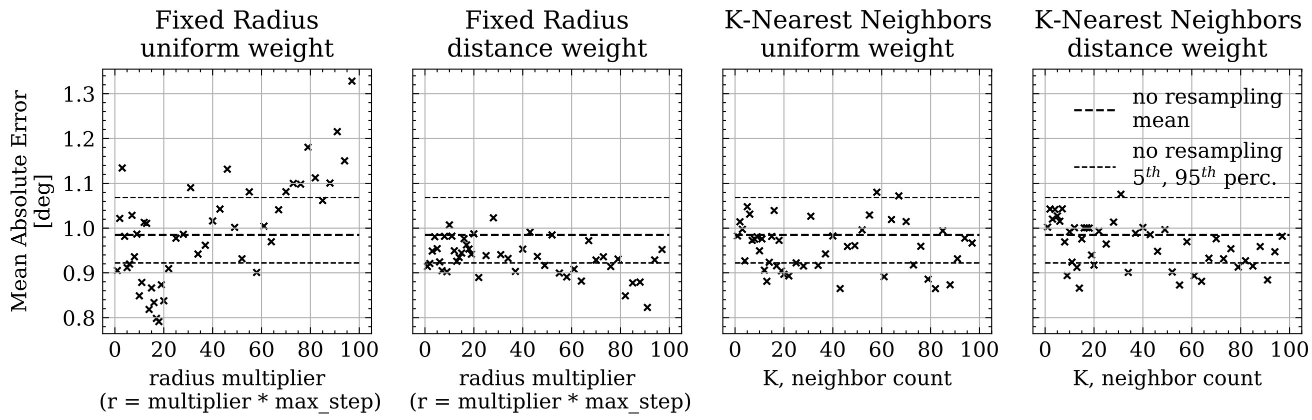


**Fig. 22.** Resampling error study, results in terms of mean absolute error.

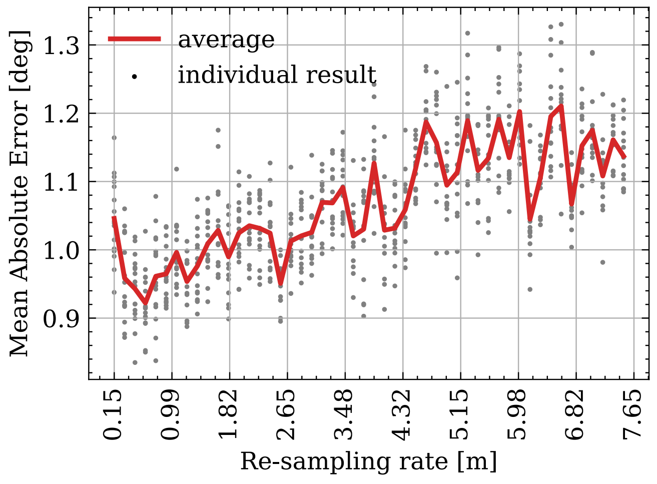


**Fig. 23.** Sampling rate study.

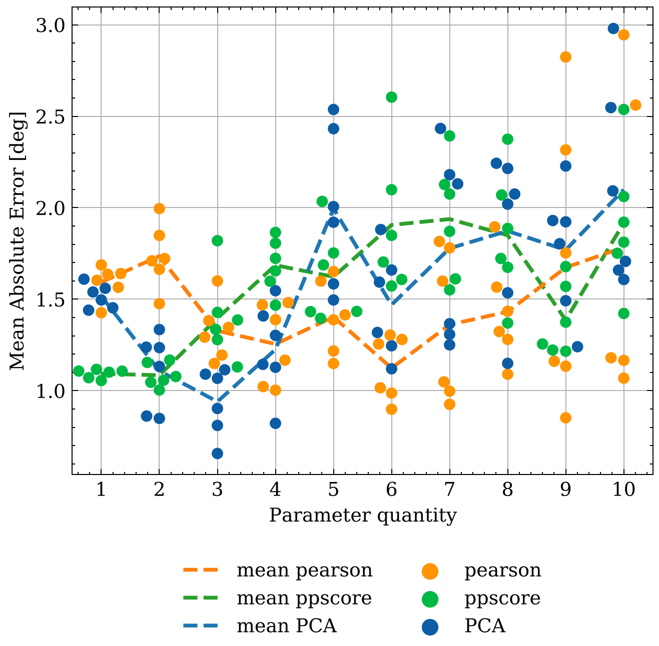


**Fig. 25.** Model selection strategy for MWD Continuous Inclination prediction based on inclination change.

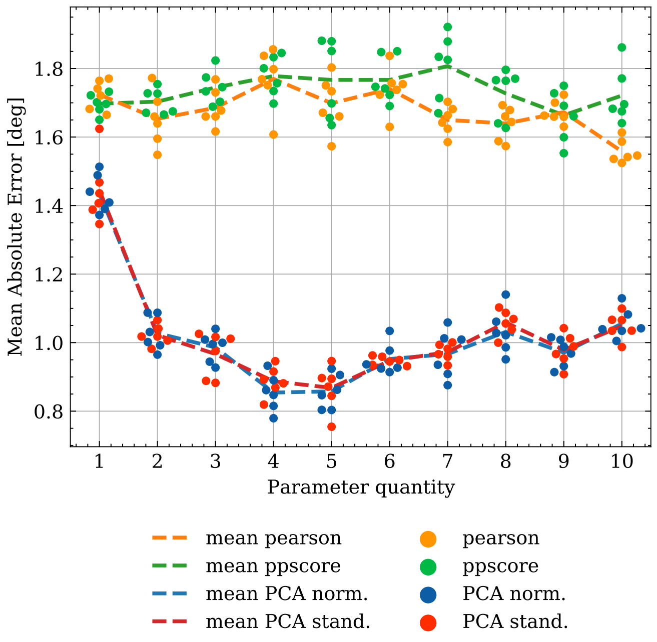


**Fig. 24.** Model selection strategy for MWD Continuous Inclination prediction.

**Table 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 |

algorithm. Out of those two the distance weighted KNN is likely a better choice, as it overall provides low error no matter the exact neighbor count selected.

### *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

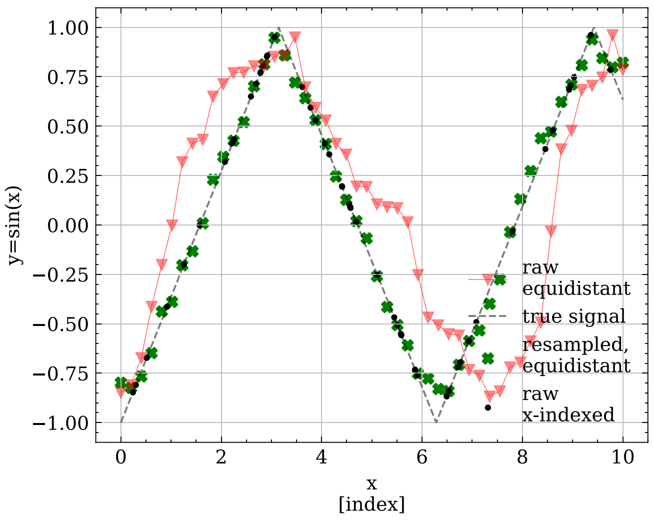


**Fig. A.26.** Example of resampling issues, triangular function.

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 (Tunkiel et al., 2021), 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 Fig. 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 F9 A, depth based dataset, from Equinor's Volve data (Equinor, 2018) converted to CSV files (Tunkiel et al., 2020d). 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.

Fig. 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 is 0.15 m. 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, 5th, and 95th 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

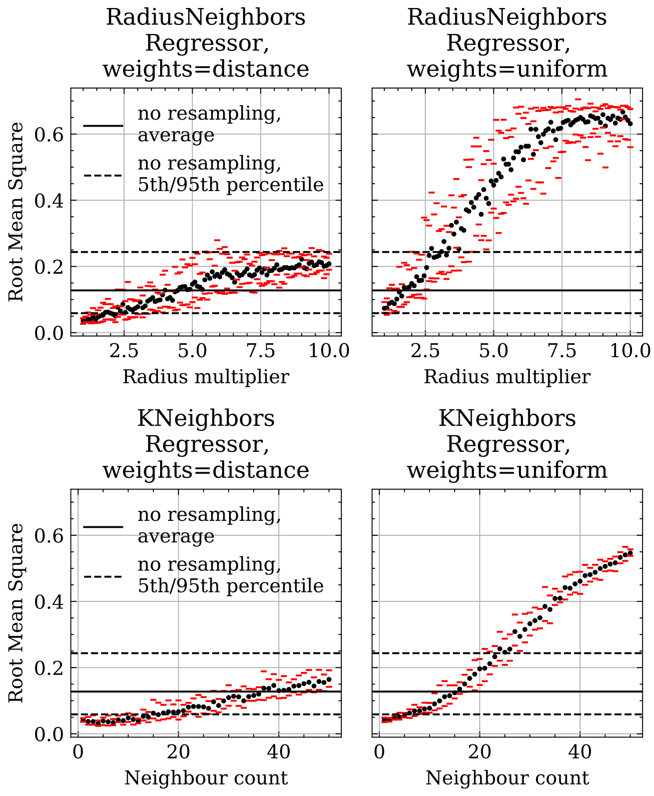


**Fig. A.27.** RMS error between resampled values and ground truth.

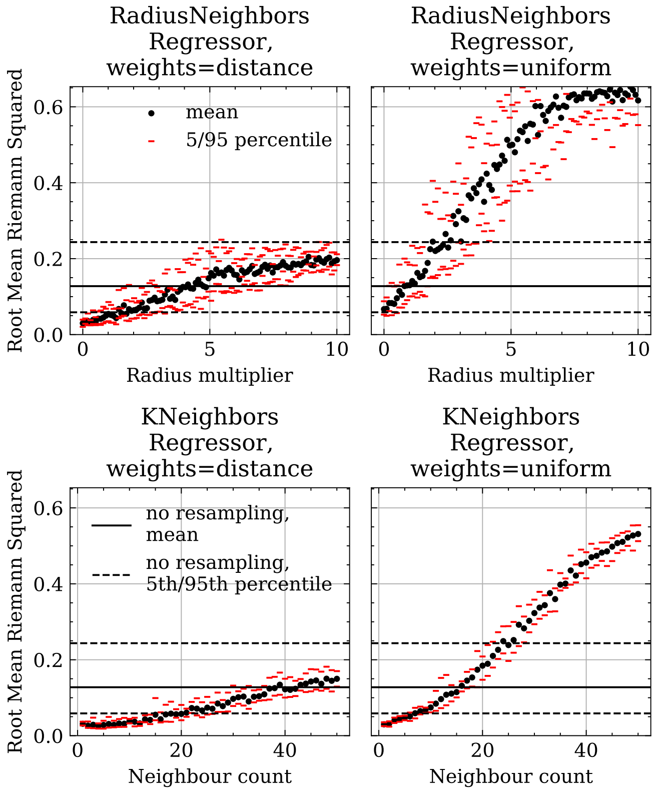


**Fig. A.28.** RMS as calculated through RMRS.

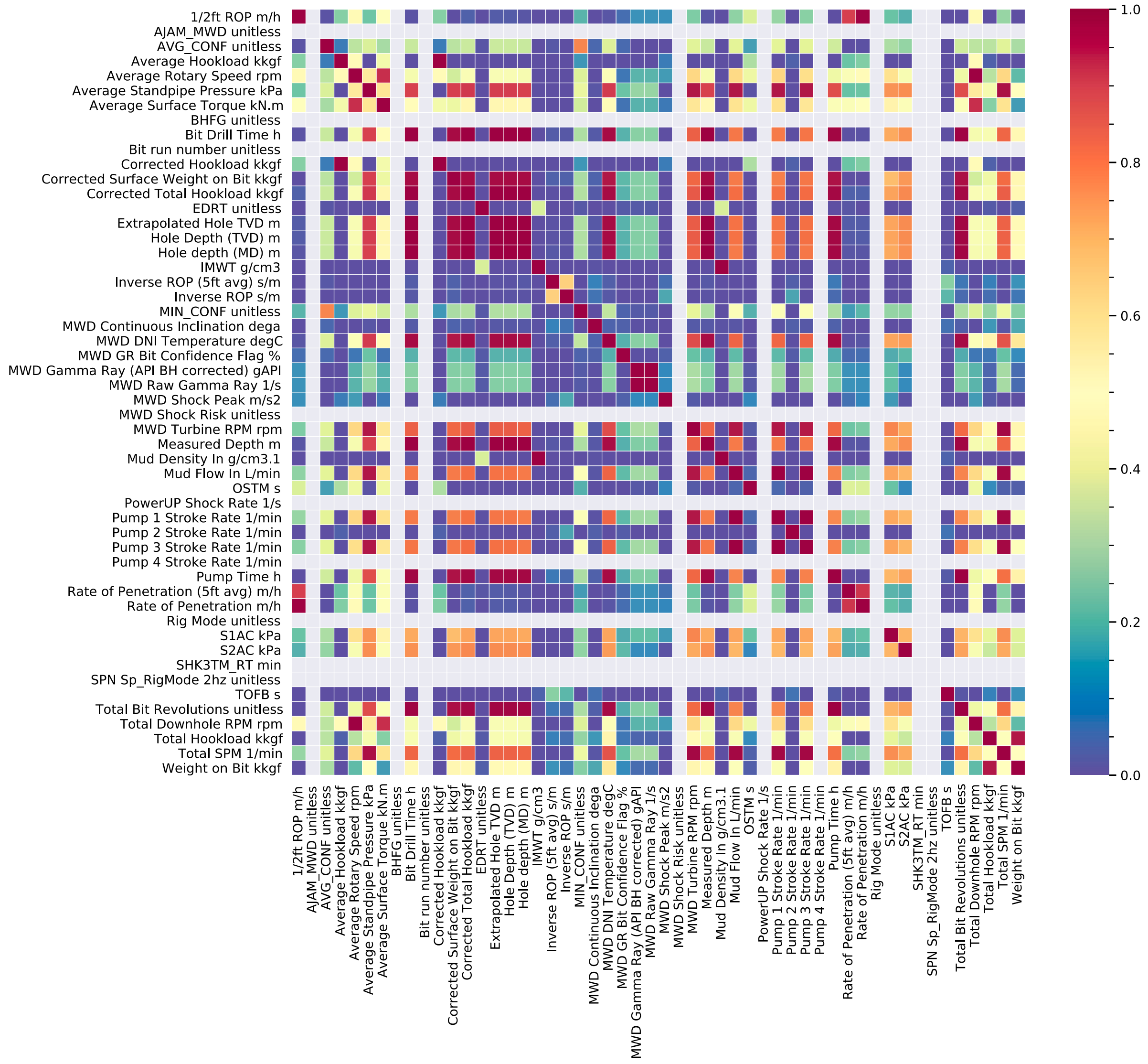


**Fig. A.29.** Volve well F9a, Pearson correlation.

points between MD=x−5 [m] and MD=x+5 [m]. This may or may not be an issue, depending on the specific application.

### 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 Fig. 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.

## 5. Attribute selection and PCA configuration

Two attribute selection strategies were implemented in the semi-automatic process presented in this paper, Pearson correlation coefficient (Benesty et al., 2009), and Predictive Power Score (ppscore) (Wetschoreck et al., 2020) 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 (i.e. *(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. Fig. 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

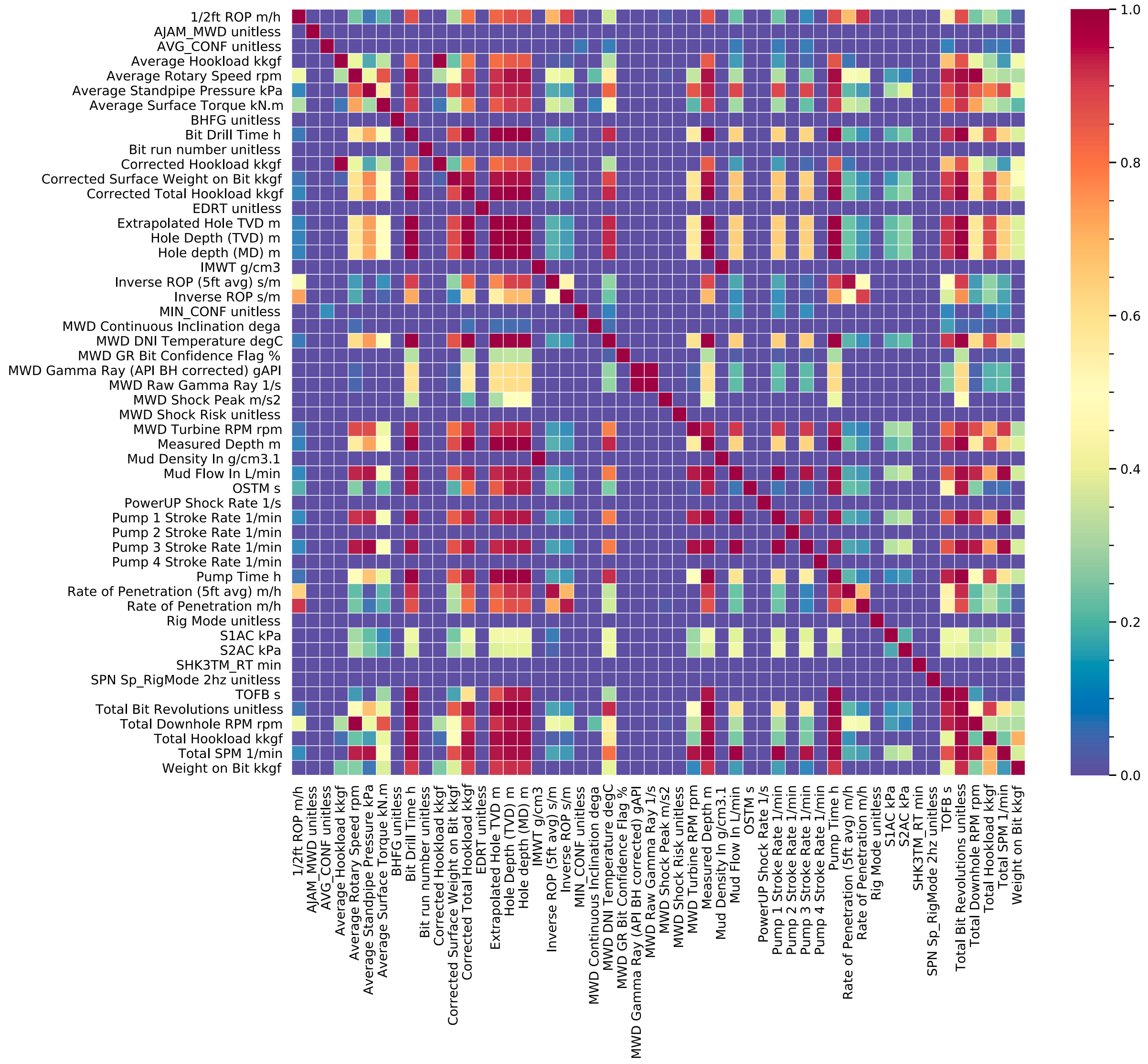


**Fig. A.30.** Volve well F9a, ppscore correlation.

quantity reduction, as it captures the behavior 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 Fig. 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, Average 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*, MWD *DNI Temperature degC*, *Extrapolated Hole TVD m*, and *Corrected Total Hookload kkgf*.

**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, Fig. A.29 for Pearson coefficient and Fig. A.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.

## 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 (Tunkiel et al., 2021), and therefore various approaches to pre-processing were likely to affect the results.

### 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 behavior.

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 Fig. 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.

### 6.2. Transformers

In 2017 a new architecture was introduced called a Transformer (Vaswani et al., 2017). 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 (Mignan and Broccardo, 2019) 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.

### 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 (He et al., 2019) or lithofacies recognition (Lima et al., 2019). 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.

### 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 (Feng et al., 2021). While the case study of predicting inclination was explored in the past to benchmark performance against simpler methods (Tunkiel et al., 2021) showing big benefits of RNNs, it is not guarantied to be the case for all problems.

## 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.

### 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 resampling rate, especially utilizing hyperparameter tuning for different settings, which is extremely demanding of computational resources. More research towards the behavior 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.

## CRediT authorship contribution statement

**Andrzej T. Tunkiel:** Conceptualization, Methodology, Software, Data curation, Writing – original draft. **Dan Sui:** Formal analysis, Writing – review & editing, Methodology, Supervision. **Tomasz Wiktorski:** Resources, Writing – review & editing, Supervision.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Appendix

See Figs. A.28–A.30.

## References

Barbosa, L.F.F., Nascimento, A., Mathias, M.H., de Carvalho, J.A., 2019. Machine learning methods applied to drilling rate of penetration prediction and optimization - a review. J. Pet. Sci. Eng. 183, 106332. http://dx.doi.org/10.1016/j.petrol.2019.106332.

Benesty, J., Chen, J., Huang, Y., Cohen, I., 2009. Pearson correlation coefficient. In: Noise Reduction in Speech Processing. Springer, pp. 1–4.

Bentley, J.L., 1975. Survey of Techniques for Fixed Radius Near Neighbor Searching. Technical Report, Stanford Linear Accelerator Center, Calif. USA.

Caflisch, R.E., 1998. Monte Carlo and quasi-Monte Carlo methods. Acta Numer. 7, 1–49. http://dx.doi.org/10.1017/S0962492900002804.

Chhantyal, K., Hoang, M., Viumdal, H., Mylvaganam, S., 2018. 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. Linköping University Electronic Press, pp. 561–567, Number: 142 tex.organization.

Chung, J., Gulcehre, C., Cho, K., Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR abs/1412.3, URL: http://arxiv.org/abs/1412.3555. tex.arxivid: 1412.3555.

Engelke, N., Sealey, V., 2009. 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.

Equinor, 2018. Volve field data (CC BY-NC-SA 4.0). URL: https://www.equinor.com/en/news/14jun2018-disclosing-volve-data.html.

Esmaeilzadeh, S., Salehi, A., Hetz, G., Olalotiti-lawal, F., Darabi, H., Castineira, D., 2019. A general spatio-temporal clustering-based non-local formulation for multiscale modeling of compartmentalized reservoirs. In: SPE Western Regional Meeting Proceedings 2019. OnePetro, http://dx.doi.org/10.2118/195329-MS.

Esmaeilzadeh, S., Salehi, A., Hetz, G., Olalotiti-lawal, F., Darabi, H., Castineira, D., 2020. Multiscale modeling of compartmentalized reservoirs using a hybrid clustering-based non-local approach. J. Pet. Sci. Eng. 184, 106485. http://dx.doi.org/10.1016/J.PETROL.2019.106485.

Feng, R., Grana, D., Balling, N., 2021. Imputation of missing well log data by random forest and its uncertainty analysis. Comput. Geosci. 152, 104763. http://dx.doi.org/10.1016/J.CAGEO.2021.104763.

Fix, E., 1985. Discriminatory Analysis: Nonparametric Discrimination, Consistency Properties. Vol. 1. USAF school of Aviation Medicine.

Geekiyanage, S.C., Sui, D., Aadnoy, B.S., 2018. 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, American Society of Mechanical Engineers., http://dx.doi.org/10.1115/OMAE2018-77510.

Geekiyanage, S.C.H., Tunkiel, A., Sui, D., 2020. Drilling data quality improvement and information extraction with case studies. J. Petrol. Explor. Prod. Technol. http://dx.doi.org/10.1007/s13202-020-01024-x.

Geller, S., 2019. Normalization vs standardization — Quantitative analysis | by shay geller | towards data science. URL: https://towardsdatascience.com/normalization-vs-standardization-quantitative-analysis-a91e8a79cebf.

He, M., Zhang, Z., Ren, J., Huan, J., Li, G., Chen, Y., Li, N., 2019. Deep convolutional neural network for fast determination of the rock strength parameters using drilling data. Int. J. Rock Mech. Min. Sci. 123, 104084. http://dx.doi.org/10.1016/J.IJRMMS.2019.104084.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9, 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735, URL: http://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735.

Kaufman, S., Rosset, S., Perlich, C., 2011. Leakage in data mining: Formulation, detection, and avoidance. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM Press, New York, New York, USA, pp. 556–563. http://dx.doi.org/10.1145/2020408.2020496.

Landau, H.J., 1967. Sampling, data transmission, and the nyquist rate. In: Proceedings of the IEEE. Vol. 55. IEEE, pp. 1701–1706.

Lima, R.P.d., Suriamin, F., Marfurt, K.J., Pranter, M.J., 2019. Convolutional Neural Networks As Aid in Core Lithofacies Classification. Vol. 7. Society of Exploration Geophysicists and American Association of Petroleum Geologists, pp. SF27–SF40. http://dx.doi.org/10.1190/INT-2018-0245.1, http://www.seg.org/interpretation, URL: https://library.seg.org/doi/abs/10.1190/INT-2018-0245.1.

Lipton, Z.C., Kale, D.C., Elkan, C., Wetzel, R., 2015. Learning to diagnose with LSTM recurrent neural networks. arXiv preprint arXiv:1511.03677.

Liu, B., 2017. Lifelong machine learning: a paradigm for continuous learning. Front. Comput. Sci. 11, 359–361. http://dx.doi.org/10.1007/s11704-016-6903-6.

Mignan, A., Broccardo, M., 2019. One neuron versus deep learning in aftershock prediction. Nature 574 (7776), E1–E3. http://dx.doi.org/10.1038/s41586-019-1582-8, URL: https://www.nature.com/articles/s41586-019-1582-8.

Neil, D., Pfeiffer, M., Liu, S.C., 2016. Phased lstm: Accelerating recurrent network training for long or event-based sequences. arXiv preprint arXiv:1610.09513.

Osarogiagbon, A., Muojeke, S., Venkatesan, R., Khan, F., Gillard, P., 2020. A new methodology for kick detection during petroleum drilling using long short-term memory recurrent neural network. Process Saf. Environ. Prot..

Pasini, A., 2015. Artificial neural networks for small dataset analysis. J. Thorac. Dis. 7, 953–960. http://dx.doi.org/10.3978/j.issn.2072-1439.2015.04.61.

Pearson, K., 1901. LIII. On lines and planes of closest fit to systems of points in space. London Edinb. Dublin Philos. Mag. J. Sci. 2, 559–572. http://dx.doi.org/10.1080/14786440109462720.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in python. J. Mach. Learn. Res. 12, 2825–2830.

Prechelt, L., 1998. Early Stopping - But when? Springer, Berlin, Heidelberg, pp. 55–69. http://dx.doi.org/10.1007/3-540-49430-8_3, URL: https://link.springer.com/chapter/10.1007/3-540-49430-8_3.

Rosenblatt, F., 1961. Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms. Technical Report, Cornell Aeronautical Lab Inc Buffalo NY.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. Nature 323, 533–536. http://dx.doi.org/10.1038/323533a0.

Sobol, I.M., 1993. Sensitivity estimates for nonlinear mathematical models. Math. Model. Comput. Exp. 1, 407–414.

Song, S., Hou, J., Dou, L., Song, Z., Sun, S., 2020. Geologist-level wireline log shape identification with recurrent neural networks. Comput. Geosci. 134, 104313. http://dx.doi.org/10.1016/j.cageo.2019.104313.

Stacklies, W., Redestig, H., Scholz, M., Walther, D., Selbig, J., 2007. PCAMethods—a bioconductor package providing PCA methods for incomplete data. Bioinformatics (Oxford, England) 23, 1164–1167. http://dx.doi.org/10.1093/bioinformatics/btm069.

Sui, D., Sukhoboka, O., Aadnøy, B.S., 2018. Improvement of wired drill pipe data quality via data validation and reconciliation. Int. J. Autom. Comput. 15, 625–636. http://dx.doi.org/10.1007/s11633-017-1068-9, URL: http://link.springer.com/10.1007/s11633-017-1068-9.

Tunkiel, A., Github for TOPPMEIS project. URL: https://github.com/AndrzejTunkiel/Tape.

Tunkiel, A.T., Sui, D., Wiktorski, T., 2020a. Data-driven sensitivity analysis of complex machine learning models: A case study of directional drilling. J. Pet. Sci. Eng. 195, 107630. http://dx.doi.org/10.1016/j.petrol.2020.107630.

Tunkiel, A.T., Sui, D., Wiktorski, T., 2020b. Reference dataset for rate of penetration benchmarking. J. Pet. Sci. Eng. http://dx.doi.org/10.1016/j.petrol.2020.108069.

Tunkiel, A.T., Sui, D., Wiktorski, T., 2021. Training-while-drilling approach to inclination prediction in directional drilling utilizing recurrent neural networks. J. Pet. Sci. Eng. 196, 108128.

Tunkiel, A.T., Wiktorski, T., Sui, D., 2020c. Continuous drilling sensor data reconstruction and prediction via recurrent neural networks. In: ASME 2020 39th International Conference on Ocean, Offshore and Arctic Engineering. http://dx.doi.org/10.1115/OMAE2020-18154.

Tunkiel, A.T., Wiktorski, T., Sui, D., 2020d. Drilling dataset exploration, processing and interpretation using volve field data. In: ASME 2020 39th International Conference on Ocean, Offshore and Arctic Engineering. http://dx.doi.org/10.1115/OMAE2020-18151.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention is all you need. In: Advances in Neural Information Processing Systems 2017-December. pp. 5999–6009, URL: https://arxiv.org/abs/1706.03762v5. Neural information processing systems foundation tex.arxivid: 1706.03762.

Wetschoreck, F., Krabel, T., Krishnamurthy, S., 2020. 8080Labs/ppscore: zenodo release. http://dx.doi.org/10.5281/ZENODO.4091345, URL: https://zenodo.org/record/4091345.

Yu, Y., Si, X., Hu, C., Zhang, J., 2019. A review of recurrent neural networks: LSTM cells and network architectures. Neural Comput. 31, 1235–1270. http://dx.doi.org/10.1162/neco_a_01199.