University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/Specialization: Industrial Asset Management	Spring semester, 2019 Open
Writer: Santiago Echeverri Duque	Santiago Echeverri Duque ………………………………………… (Writer's signature)
Faculty supervisor: Prof. Jayantha P. Liyanage **External supervisor:** Prof. Zaid Al-Ars	
Thesis title: Evaluation of machine learning for optimization and anomaly detection in offshore drilling operations.	
Credits (ECTS): 30	
Key words: Machine learning, rate of penetration, anomaly detection, optimization.	Pages: 81 Stavanger, 10.07.2019

Machine Learning FOR Offshore Drilling Operations

Santiago Echeverri Duque

July 2019

UNIVERSITY OF STAVANGER AND DELFT
UNIVERSITY OF TECHNOLOGY

MASTER THESIS

Evaluation of machine learning for optimization and anomaly detection in offshore drilling operations.

Author:
Santiago ECHEVERRI

Supervisor:
Prof. Jayantha P. LIYANAGE
and Prof. Zaid AL-ARS

A thesis submitted in fulfillment of the requirements
for the degree of Master

in the

Industrial Asset Management

Department of Mechanical and Structural Engineering and Materials Science

Cover adapted from [66]

July 10, 2019

UNIVERSITY OF STAVANGER AND DELFT UNIVERSITY OF TECHNOLOGY

Abstract

Department of Mechanical and Structural Engineering and Materials Science
Industrial Asset Management

Master

Evaluation of machine learning for optimization and anomaly detection in offshore drilling operations.

by Santiago ECHEVERRI

Studies of drilling operation have been focused on control parameter optimization to improve the rate of penetration and mechanical specific energy. Drilling is a complex operation with uncontrolled parameters and disturbances that could generate non-productive time during the bit trajectory. The operation is not fully automated increasing the likelihood of misbehavior events. A possible scenario to improve the operation is proposed in this research, coupling predictions of parameters with anomaly detection algorithms to minimize the cost of the drilling operation. It means that at the same time that we are optimizing we need to issue an alert in case of misbehavior. Machine learning algorithms have contributed to both approaches uncovering relations between input parameters and the quantity of interest. This research is conducted based on the following structure: First, machine learning models have been implemented with incremental training data available to predict the rate of penetration. Second, detection of misbehavior models of control, uncontrolled and response parameters have been integrated into the algorithm. Our experiments showed that random forest is a competent machine learning algorithm to predict the rate of penetration with a performance error (root mean squared error) of 2,92 m/hr (9,57 ft/hr) in static analysis and 4,43 m/hr (14,55 ft/hr) average error increasing the availability of data. Furthermore, isolation forest represents a flexible method detecting anomalies in the context of unsupervised learning. Both methods, random forest and isolation forest, performance under a similar structure with incremental data architecture. Algorithms for anomaly detection exposed between 52 and 69 anomalies over 6511 points. Results indicated that just one method could miss the detection of a critical event. Finally a virtual detector is proposed with an architecture of five layers to optimize drilling operations.

Keywords: Machine learning, anomaly detection, rate of penetration, drilling

Acknowledgements

The opportunity to be abroad has brought me multiple unequaled experiences. Every moment in Norway (University of Stavanger) and The Netherlands (Delft University of Technology) has taught me about tolerance, friendship, happiness, resilience, and effort. In my professional career, I have always undertaken challenges, and this process during my master thesis has not been an exception. Data science has captured all my attention, my time and my readings. Therefore, I decided to complement my experience and academic path in this field. It is time to give thanks to all that have supported me during this process.

My gratitude to Zaid Al-Ars, from Delft University of Technology, for welcoming me in his research team of Quantum & Computer Engineering. Every meeting had a friendly conversation before substantial machine learning questions. His office was always open for basic and complex questions. During his constrained agenda, he was always available to have meetings with companies interested in our topic. His guidance in technical and strategic contributed to this result.

My gratitude to Jayantha P. Liyanage, from the University of Stavanger, for his instructions during the project. Although we work from a distance, our frequent meetings were valuable to achieve the correct approach. His promptitude in every email contributed to improving our outcome. Every observation for this research was valuable.

My family, eternally my center and my company even from the distance. A massive thanks to my parents that have supported me in every challenge that I have decided to undertake. At the same time, thanks to all of my friends and roommates in Norway and The Netherlands for their time inside and outside the university.

Santiago Echeverri Duque
Delft, July 2019

Contents

List of Figures

List of Tables

List of Abbreviations

AdaGrad	**A**daptive **G**radient
ANN	**A**rtificial **N**eural **N**etwork
ADABR	**A**daptive **B**oost **R**egression
BHA	**B**ottom **H**ole **A**ssembly
CV	**C**ross **V**alidation
DNN	**D**eep **N**eural **N**etwork
DTW	**D**ynamic **T**ime **W**arping
DT	**D**ecision **T**ree
ECD	**E**quivalent **C**irculating **D**ensity
GAN	**G**enerative **A**dversarial **N**etwork
HLN	**H**idden **L**ayer **N**euron
IF	**I**solation **F**orest
KNR	**K** **N**earest **R**egression
LR	**L**inear **R**egression
LOF	**L**ocal **O**utlier **F**actor
LSTM	**L**ong **S**hort **T**erm **M**emory
LBFGS	**L**imited memory **B**roydenn **F**letcher **G**oldfarb **S**hanno
MDI	**M**ean **D**ecrease **I**mpurity
ML	**M**achine **L**earning
MLP	**M**ulti **L**ayer **P**erceptron
MSE	**M**echanical **S**pecific **E**enery
NN	**N**eural **N**etwork
NPT	**N**on **P**roductive **T**ime
RBF	**R**adial **B**asis **F**unction
RSS	**S**tick **S**lip **R**atio
RF	**R**andom **F**orest
RMSE	**R**oot **M**ean **S**quare **E**rror
ROP	**R**ate **O**f **P**enetration
RPM	**R**evolutions **P**er **M**inute
ReLU	**R**ectifier **L**inear **U**nit
SAX	**S**ymbolic **A**ggregate **A**pproximation
SGD	**S**tochastic **G**radient **D**escent
SVM	**S**upport **V**ector **M**achine
SVR	**S**upport **V**ector **R**egression
TOB	**T**orque **O**n **B**it
WOB	**W**eight **O**n **B**it
WITSML	**W**ellsite **I**nformation **T**ransfer **S**tandard **M**arkup **L**anguage
XGB	**E**xtreme **G**radient **B**oosting

Physical Constants

Speed of Light	$c_0 = 2.997\,924\,58 \times 10^8\,\mathrm{m\,s^{-1}}$ (exact)

List of Symbols

q	flow	$liters/min$
A_b	cross sectional area of bit	m^2
ω	angular frequency	rad

All my work is dedicated to my parents and siblings who inspire, support and teach. I could not have done without them…

Chapter 1

Introduction

Selecting operational parameters that maximize a desirable measure of drilling performance is one of the goal of drilling optimization [58]. Efforts are geared to vary drilling parameters and achieve minimum mechanical specific energy and optimum penetration rate [5]. Drilling parameters measured on the rig can be classified into control parameters, uncontrollable parameters, and response parameters [30]. Control parameters can be controlled by the drilling engineer on the rig: weight on bit, drilling rotational speed, and drilling fluid (mud) flow rate [30][34][58]. Strength of the rock, geological properties, maximum pump power correspond to uncontrollable parameters which cannot be changed by engineers while drilling a well. Response parameters (the objectives) are those which change when control parameters are changed: rate of penetration, mechanical specific energy, downhole vibrations, and torque on bit. Still, there are several variables which are difficult or impossible to measure on real-time [58].

Drilling is a complex operation with uncontrolled parameters and disturbances that could generate non productive time during the bit trajectory. Many variables profoundly influence the rate of penetration which include, but are not limited to, parameters on the surface, formation properties such as rock strength, abrasiveness, heterogeneity, pore pressure and permeability, parameters on the surface, bit design, mud, human factors, downhole conditions, and mud rheology [34], hole diameter, hole cleaning and hydraulics[58]. Any incorrect decision, deviation, or unknown scenario could interfere in the normal operation. The operations are not fully automated increasing the likelihood of misbehavior events [27]. The efficiency of such operations depends on the driller skills.

This research evaluates machine learning models to contribute in the parameter optimization and anomaly detection. On one hand, machine learning models have been implemented to predict rate of penetration [34] [58] [56]. These models leverage statistics to uncover relations between any prescribed inputs (features/predictors) and the quantity of interest (response) [58]. With no set equation, machine learning model allowance segmentation of the drilling operational parameter space. However, the increased model complexity reduces interpretability of how and adjustment to the inputs will affect the output. On the other hand, machine learning models for anomaly detection has supported the predictability of failures events [20] specially in predictive maintenance of equipment [35]. For humans, it is difficult to recognize abnormal state and normal state using raw data [53]. Training machine learning models to learn the normal state and identify the divergent pattern is valuable.

The main goal of this study is the characterization of machine learning models for prediction of response parameters and anomaly detection. Therefore, the research is conducted based on the following structure: First, machine learning models have been evaluated with incremental training data available to predict rate of penetration. Second, detection of misbehavior models of control, uncontrolled and response parameters have been integrated to the algorithm. It provides drilling superintendent, drilling supervisors and real time operations centers staff with better tool for decision making. Any optimization or detection of misbehavior represent saving cost, improve safety and increase efficiency in drilling operation and add value to the investment of operational centers, monitoring and management systems.

The present document is distributed in 6 chapters. Chapter 1 includes the scope, objectives, limitations, and methodology of the research. In Chapter 2, the current machine learning methods and tools applied in drilling operations are reviewed during the revision of academic literature. Then, the relevant alternatives based on the literature review are defined and analyzed in Chapter 3. The implementation architecture is structured in Chapter 4. The experimental results and analysis are included in Chapter 5. The last but not least, the conclusion and future research are presented in Chapter 6.

1.1 Scope

Different machine learning models has been used in the oil and gas industry, most of them focused on prediction of rate of penetration. Looking forward to understand the implementation of models already used in previous studies and complement the analysis with additional strategies, in this research the scope will be to evaluate machine learning and optimization methods to predict response parameters (rate of penetration) and detect misbehavior in drilling operations. To explain with more details the general scope, the specific goals are:

1. Machine learning method analysis:
    - Evaluate relation training test set required to obtain predictability with the lowest error.
    - Investigate the feature set required to train the machine learning model.
    - Identify methods that allows predictability with the lowest error.
2. Detection of misbehavior:
    - Select the machine learning model to detect misbehavior with lowest false positive and false negative flag.

1.2 Objectives

Based on the scope described in previous section, the objectives with the respective metrics are defined as follow:

1. Characterization of machine learning model with stationary analysis and incremental training data available to predict rate of penetration.
    - Select optimal hyperparameter combination (hyperparameters)
    - Determine relevant features (features with 95 % of importance)
    - Identify model with best performance (RMSE and run time)
    - Analyze training to test set ratio (% training/test)
    - Assess incremental data availability strategy using batches per formation (RMSE and run time)
2. Identification of misbehavior of operational parameters (control, uncontrolled and response parameters)

- Test machine learning methods to identify anomalies in stationary analysis (number of anomaly detection)
- Test machine learning methods to identify anomalies in streaming data (number of anomaly detection)

1.3 Limitations

This research is governed by some limitations, which are:

- The purpose of the thesis embraces the evaluation of the machine learning models. Therefore, the pipeline is not implemented in real-time operations. Data preparation, modeling, evaluating, model selection and optimizing are include in the development of the analysis. However, operationalizing the models in a production environment are not included in our process. Further code developments and use of cloud solution are required to reach that goal.
- Simulation of incremental data available reproducing the real time collection of data is not included in the research. It means, the algorithm does not reproduce the collection of data per second or minute or subsequent scales.
- Open source services are used during the stage of experimentation and implementation.
- The information submitted here does not have any connection with Utah Forge U.S Department of Energy or University of Utah.
- The geological formation identified for Well 58-32 and Well 21-31 allowed us to delimited the top of each data set for batch analysis. However, geological analysis or interpretation is not included in the present research.
- The data set available does not include labels of anomalies. It means that metrics like confusion matrix to identify the best model of anomaly detection are not included in this research.

1.4 Methodology

With the research methodology designed for this project, we are exploring the following questions: *What have been the machine learning techniques used for prediction and anomaly detection in drilling operation?*, *What are the machine learning algorithms that fit with our context of drilling operation data sets?*. *How appropriate is the performance of the models?*, and *Is there an additional model to predict variables and detect anomalies?*. To achieve an interpretation for the previous questions, we are going to describe our approach:

- **Literature review**: This review will include the analysis of machine learning models implemented in drilling operation targeting the prediction of rate of penetration. How to split the training and test set will be inspected. Identification of physics-based models used in well drilling planning or real-time optimization will be required to understand and compare the main features of analysis. Possible optimization algorithms should be explored to find optimum parameters. Objective functions must be identified to analyze the impact of the parameter variations. Therefore, how to couple training data, predictions, objective functions and optimization of parameters will require a deep analysis in previous work. This structure will contribute to analyze the feasibility of the machine learning models to optimize operational parameters. The last but not least, anomaly detection techniques will be explored in this stage.

- **Alternative solutions and selection**: Based on literature review, the relevant machine learning models, physics-based models for objective function, and anomaly detection models will be defined. The study will include the analysis of main hyper-parameters, and advantage and disadvantage for each method. The outcome of this analysis will be the selection of the respective models. Parallel to this activities, preliminary experiments will be developed to inspect the functionalities and performance of models.
- **Implementation**: The technological configuration will be implemented based on the models and algorithms selected. Coding tasks are developed during this stage. Here, the important task is to compare the performance of the models. Different open source environments are available to implement the code. We are going to explore the different alternatives to achieve the best alternative for our purpose.

What cases of study are this research going to analyze? Specifically, in the activities of experimental results and evaluation (Chapter 5) we are going to use the cases of study disclosed of Utah Well 58-32 [46] and Fallon Well 21-31 [54].

- The data set Well 58-32 Milford, Utah (USA) [46] contains processed drilling data with the following characteristics.
  - Well 58-32 was drilled vertically in 58 days to a depth of 7,536 ft.
  - The intent of the drilling was to determine the characteristics of the rock within the target formation and at the depth and temperatures of interest.
  - Information about geological formation and stratigraphy of the well is found in Frontier report [23].
- The data set Well 21-31 Fallon, Nevada (USA) [54] contains: *well lithology logs* and *well logging data*.
  - *Well lithology log* fields include: geologic unit, depth from, depth to, unit thickness, unit thickness, and full unit name.
  - *Well logging data* includes: daily reports, well logs (drill rate, lithology, fractures, mud losses, minerals, temperature, gases, and descriptions), mud reports, drilling parameter plots, daily mud loss summaries, survey reports, progress reports, plan view maps (easting, northing), and wireline logs.
  - Well 21-31 was drilled in 25 days (February 2018) to a depth of 6,108 ft.

The methodology proposed by Theodoridis et al [62] will be used in our implementation during the machine learning analysis for prediction and anomaly detection: *pre-processing*, *feature selection and/or reduction*, *regression design* and *anomaly detector design* and *system evaluation*.

1.5 Assistance and design tools

Google Colaboratory allows to develop the coding task of machine learning models. Consequently, the following packages will be used: Scikit-Learn for machine learning algorithms [48], Pandas [43] for data extraction and preparation and, Matplotlib [37] and Seaborn for data visualization as well as Bokeh [13] for interactive visualization, and Keras [22] and Tensorflow [1] for deep learning algorithms.

The following online examples [7], [18], [16], [39], [17] will guide during the coding stage.

At the same time, this project will be developed in the research group Quantum and Computer Engineering from TUDelft. Therefore, the thesis developed by Helmiriawan [35] and Hes [36], who are part of this team, will contribute with the structure, application and theory for this project.

1.6 Results and outcomes

The outcomes of the research project are defined as follow:

1. In machine learning method analysis, the result will be the characterization of machine learning models to predict response parameter (rate of penetration).
2. In detection of misbehavior, the development will be the characterization of machine learning models to detect misbehavior in drilling operation.

Highlights:

- The data set available to develop the research comes from Well 58-32 Milford, Utah (USA) [46] and Well 21-31 Fallon, Nevada (USA) [54]
- The outcome of the researches is focused in the characterization of machine learning models for rate of penetration prediction and anomaly detection.
- Google Colaboratory will be the main tool to develop the implementation of machine learning models. This tool offers easy access to many packages available for machine learning and deep learning. Bokeh application will be used for interactive visualization.
- The project is developed in Quantum and Computer Engineering research group from TUDelft.

Chapter 2

Literature review

2.1 Drilling operation

Oil and gas industry is divided in *upstream*, *midstream*, and *downstream.* In *upstream* we have exploration, drilling and production of crude oil and natural gas. Meanwhile, processing, storing, transporting and marketing of oil, natural gas and natural gas liquids are part of *midstream.* On the other hand, *downstream* corresponds to refining, processing, and purifying, marketing and distribution of products derived from crude oil and natural gas. In this project we are going to focus on **drilling operations**. According to Speight [59], drilling is the most essential activity in oil and gas recovery. Drilling accounts for a significant part of oil and gas budgets [29] [9]. The costs of rig operations make up a significant part of the drilling expense [65]. Therefore, optimization is a central priority of all operators. The types of drilling operations include drilling, circulating and moving the drillstring up or down [27].

The first stage in the extraction of crude oil from an underground reservoir is to drill a well into the reservoir [59]. It is only through the actual penetration of the formation by the drill bit that the presence of recoverable crude oil and natural gas can be confirmed. We can find drilling operation onshore and offshore. Drilling operations are more cost efficient when rate of penetration are optimized [33], representing how fast or slow a well is being drilled. Efforts are geared to vary drilling parameter to achieve minimum *mechanical specific energy* and obtaining optimum *penetration rate* as formation strength are reasonable uniform within the same formation interval [5]. Drilling offshore wells are drilled by lowering a drill string consisting of a drill bit, drill collar, and drill pipe through a conduit (riser) that extends from the drilling rig to the sea floor. Some of the main characteristics of drilling operation are described as follow [59]:

- Drill bits have cones with teeth and are designed to break the rock by indention and a gouging action. As the cones roll across the bottom, the teeth press against the formation with enough pressure to exceed the failure strength of the rock at which rock fracture occurs.

- At the surface, a rotary table turns the drill string and the drill bit teeth penetrate the sea floor sediment and the various rock formations that overly the reservoir while a drilling fluid is pumped into the drill pipe from a tank on the surface and the mud flows through perforations in the drill bit. The weight of the mud exerts a pressure greater than the pressure in the rock formations, and, therefore, keeps the well under control.

- As the drill bit penetrates further into the rock formations, strings of steel pipe (casing) are run into the well and cemented into place in order to seal off the walls of the well and maintain the integrity of the well by preventing collapse of the walls.

The related parameters of drilling rig and bit, formation, and fluids include [56]:

(a) **Rig and bit parameters**: Weight on bit, torque, rotary speed (rotations per minute of the drill bit), flow rates (drilling mud), pump stroke speed, pump pressure, hook load, bit wear, type of the bit.

(b) **Formation parameters**: Local stress, hardness, mineralogy, porosity and permeability, formation abrasiveness, drillability, depth, temperature, unconfined compressive strength.

(c) **Drilling fluid properties**: Mud weight, viscosity, filtrate loss, solid content, gel strength, mud pH and yield point.

Input parameters such as weight on bit, rotations per minute of the drill bit, flow rate of the drilling mud and unconfined compressive strength of the rock [58][29] are considered relevant in drilling operational analysis. In addition to bottom hole pressure and temperature, pump pressure, torque, hole depth and bit depth [26].

2.2 Machine learning

The result of running the *machine learning algorithm* can be expressed as a function $y(x)$ which takes a feature x as input and that generates an output vector y, encoded in the same way as the target vectors [12]. The precise form of the function $y(x)$ is determined during the training phase, also known as the *learning phase*, on the basis of the training data. Once the model is trained it can then determine the identity of new feature, which are said to comprise a test set. The ability to categorize correctly new examples that differ from those used for training is known as *generalization* [12] [64]. In genera, implementing machine learning algorithms includes preprocessing, feature generation, feature selection and reduction, classification or regression, and system evaluation [62] as was mentioned for our methodology.

When the aim is to assign each input vector to one of the activities in drilling operation, we would have a *classification* problems. And it is called regression, if the desired output consists of one or more continuous variables. For instance, prediction of rate of penetration in oil and gas in which the inputs consist of the weight on bit, rotary speed and flow, we would have a regression problem. Applications in which the training data comprises examples of the input vectors along with their corresponding target vectors are known as *supervised learning* problems [12]. On the other hand, when the training data consists of a set of input vectors x without any corresponding target values. We have *unsupervised learning*, the goal in such a problem may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization.

Classical learning with supervised techniques include *linear regression*, *logistic regression*, *k-nearest neighbor*, *support vector machine* [55], *decision tree* [51], *random forest* [14], *Neural networks* [12] and advances in *deep learning* [28]. In the case of unsupervised learning we can find *clustering* [62] with k-means, hierarchical cluster analysis, and expectation maximization, *visualization and dimensionality reduction* with principal component analysis and t-distributed stochastic neighbor embedding.

2.3 Machine learning fro drilling operation

Machine learning techniques are constructed based on mathematical models. These models are able to learn the trend or behavior of experimental or real data and thus discern a pattern [9]. The main objective of seeking smart machine methods is to predict the occurrence of some problems based on previous experience with reasonable cost and time. The reliability of the method depends on the accuracy of prediction and the error between the actual and the predicted class labels of the problem. Machine learning techniques are able to find patters usually with nonlinear behaviours and multidimensional parameters.

Currently, statistical learning methods and automation can improve drilling efficiency reducing non-productive time (NPT), and decreasing the cost of drilling [65]. With no set equation, machine learning models permit segmentation of the drilling operational parameter space [58]. Different techniques have been used to optimize operations and detect anomalies. On one hand, least squares regression, random forests [65], trees, bagging and random forest [32], neural networks [58] have been used to optimize operations. When we talk about optimization, most of the studies have been focused on **rate of penetration** [26]. On the other hand, the approach of early detection of drilling events have been proposed under artificial neural networks and support vector machine.

2.3.1 Optimization of parameters, machine learning and drilling operation

Which techniques and what are the characteristics of the experiments developed in previous research? In this section, we developed a literature review about papers related to drilling operation and machine learning. The summary is displayed in Table 2.1 with the methods, goal and the respective reference.

Wallace et al. [65] used random forests, neural networks and ensemble techniques to build a predictive model for *rate of penetration*. In this study, each rock formation type is analyzed separately. Surface while drilling data was used in their implementation, and does not depend on any down hole measurements, bottom hole assembly characteristics, or formation properties for its analysis. Selecting the data, the first 500 ft of the lateral well was used as the training data, using the following 5000 ft as validation data. The model were analysed with *root mean square error* metric, obtaining an error around 23 ft/hr. By allowing the traditional physical models (Bingham, Bourgoyne and Young, Hareland and Rampersad, and Motahhari) to recalibrate the empirical coefficients every 50 ft of depth, the accuracy was improved to around 18 ft/hr *root mean square error*. According to Wallace et al. [65], this practice of recalculating the empirical coefficients every 50 ft is merely a forced overfit of the traditional model.

Searching for to optimize the ROP, Hegde et al. [32] used decision trees, bagged trees and random forests. Decision trees provide easy interpretability and hence are favored over other non-linear techniques. However, decision trees can result in substantial overfitting. This shortcoming was rectified using bagging or random forest methods to substantially increase accuracy. The analysis was developed with a training and validation set with a subsection of first 500 ft. The test set was developed with 2000 ft yet to be drilled. The data set was obtained from Tyler formation (Williston Basin of Western North Dakota). The evaluation was established in each formation, nine (9) formations in total. *Regression tree* obtained 35 ft/hr *root mean square error*, *boosting* method 34 ft/hr and *random forest* 7.4 ft/hr. Decision trees carries high variance. Having high lithology variation, decision trees have disadvantage. Averaging models helps to reduce variance where ensemble models have advantages. An additional study developed by Hegde et al. [29], *random forest* achieved an accuracy (R^2) of 0.96 and *linear regression* 0.42. This study included training, validation and test set of the data set obtained from Tyler formation. Cross validation was implemented in this analysis. Rate of penetration was predicted using weight on bit, rotary speed, flow rate and unconfined compressive strength of rock which can be manipulated by an engineer to change the mentioned rate. In an additional study, Hegde et al. [30] used *random forest* to predict rate of penetration in 12 different formations using the same features as the previous research. At each iteration, 40% of the data points were selected in a formation for training set and 60% for test set. The technique achieved to improve 28% on average the rate of penetration.

Moreover, Soares et al. [58] used random forests, support vector machines and neural networks to predict rate of penetration. Weight on bit, rotations per minute of the drill bit (RPM), and drilling fluid (mud) flow rate were the features selected for the analysis in 19 formations. The implementation included cross-validation and the input data from Williston Basin dataset (USA). For support vector machine and neural networks models, the data were standardized to zero mean and variance equal to one. The system obtained 12.58 *normalized root mean square error* with random forest. The best hyperparameters were 25 trees and 2 maximum features per split in *random forest*. In *support vector machine*, Soares et al. [58] achieved best performance with Gaussian kernel function, epsilon(ϵ) 1, penalty parameter C of the error term equal to 100, and kernel coefficient (λ) 0.1. In the neural networks grid, the solver with best score was the limited memory Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimizer, two hidden layers with 4 and 2 neurons in each one, logistic as activation function and 0.0001 in I_2 regularization (α).

With the goal to improve the efficiency of directional drilling, Pollock et al. [50] used neural network with reinforcement learning methods. Training, test and validation set were splitted. The data set contains 377,000 time points used for training and testing and 214,000 time points for network validation. Much of the information was recorded in the drilling logs and includes differential pressure, rotary torque, hook load, tool face angle, and rate of penetration as well as planned and estimated actual wellbore trajectory, weight on bit, flow rate, rotary speed and top drive center position and torque. Hierarchical clustering based on column was used to determine closely related categories, such as rotary torque and rotation speed, and which categories should be used as outputs and inputs for training the artificial neural network. In this pipeline, a generative adversarial network (GAN) was used for unsupervised learning with long short-term memory (LSTM) to automatically identify sliding time-frames and provide a framework for sliding identification for artificial neural network training. A feed-forward multilayer perceptron network was constructed as the first step toward deep machine learning for directional drilling. The neural network predicts future differential pressure and rotary torque based on current tool face, weight on bit, total pump output, rate of penetration, current rotary torque and current differential pressure. Reinforcement learning was used to refine the neural nets of the directional drilling system based on the results of an appropriate drilling simulator. After 1,800,000 training steps, normalized percentage error for differential pressure prediction was down to 0.21% and that for rotary torque prediction was at 2.72% when tested against directional drilling not included in training or validation processes.

On the other hand, Shi et al. [56] concentrates on ROP estimation using bit type and its properties, mud type and mud viscosity, formation parameters such as rock strength, formation drillability, and formation abrasiveness, and some critical drilling equipment operational parameters such as pump pressure, weight on bit, and rotary speed based on the previous drilled wells data with the extreme learning machine (ELM) and upper-layer-solution-aware (USA) model. ELM is a fast algorithm for single hidden-layer feedforward neural networks. The developed ELM and USA model are shown to be efficient (accuracy and running time) compared to traditional artificial neural network models. The training subset was 75% of the total data and testing subset correspond to 25%. The total number of target wells is summed up to 5500. The study developed by Shi et al. [56] contributed with a literature review of neural networks used for rate of penetration prediction.

2.3.2 Early detection in drilling operation and machine learning

Some of the well known misbehavior in drilling operation the other hand are stuck pipe [9], kick and fluid loss [63], severe vibration, sudden equivalent circulating density changes, high stick/slip, and severe whirling [67]. These events cause the decrease of drilling efficiency, increase of drilling cost, and even equipment failures. For example, the severity of stick/slip events can be inferred from downhole gyro data or the stick/slip ratio (SSR) derived from downhole RPM. Stick/slip with high severity can lead to reduced ROP and premature bit failure, downhole tool failure, and bottom hole assembly (BHA) failure [67].

Al-Baiyat et al. [9] proposed artificial neural networks and support vector machine to predict stuck pipe occurrences. Stuck pipe has been recognized as one of the most challenging and costly problems in the oil and gas industry [9]. The study involved classifying stuck pipe incidents into two groups - stuck and non-stuck - and also into three subgroups: differentially stuck, mechanically stuck, and non-stuck. It means, stuck pipe is considered as the dependent variable while the drilling parameters are considered the independent variables. The implementation used *sigmoid* and *tanh* as activation function. The system included 18 neurons in the input layer, 19 neurons in the hidden layer and 1 output. Meanwhile, support vector machine was implemented with linear and radial kernel. For this study, 48 data sets were used for training and 18 data sets for testing. Support vector machine was more convenient than artificial neural networks since they need fewer parameters to be optimized. The accuracy of support vector machine was over 85%.

When a drilling rig find that the mud hydrostatic is lower than the pressure within the drilled rock called kick or experiences lost circulation, it is both dangerous and expensive. The earlier these events are detected the sooner the crew can take critical corrective action, minimizing both the danger and cost associated with the event. As a kick enters the wellbore and begins making its way to the surface it shows up as a gain in the volume of mud at surface and also an increase in mud flow rate out of the well. Conversely, lost circulation occurs when some of the drilling mud is lost down hole. The early detection of kicks and loss circulation have been studied by Temizel et al.[61] where machine learning has been proposed as an alternative to define adaptive alarms thresholds. This work undertook machine learning techniques not specified in their study, which reduced false alarm rates while increasing the probability of detection. The research achieved 33 influxes detected at a gain of $0.4m^3$ in mud volume and 3.5% increase in flow. All 20 losses were detected with a loss of $1.0m^3$ and 3.5% decrease in flow. Meanwhile, Zhao et al. [67] developed a method to detect the precursors of drilling events based on drilling data such as surface data, wellbore geometry data, lithology (formation characteristics), and downhole measurements from various downhole tools. The drilling events refer to a behavior of the drilling system detected or recorded, such as severe vibration, stuck pipe, fluid loss, sudden equivalent circulating density (ECD) changes, etc. Based on various machine learning techniques, the method learn the changing trend of drilling parameters when the drilling events happen. The time series of drilling parameters were represented by symbolic aggregate approximation (SAX). The patterns of these SAX strings are clustered by unsupervised learning and then used for pattern recognition with dynamic time warping (DTW). The searching pattern recognition was proposed to classify the changing trend of drilling parameters.

The summary of machine learning technique with the respective reference is presented in Table 2.1.

TABLE 2.1: Literature review of machine learning and drilling operation

Ref.	**Goal**	**Linear regression**	**Least square regression**	**Random forests**	**Ensemble**	**Decision trees**	**Support vector**	**Neural networks**	**Symbolic aggregate approximation**
[65]	ROP prediction		X	X	X				
[33]	ROP prediction			X		X			
[29]	ROP prediction	X		X					
[30]	ROP prediction-optimization			X					
[58]	ROP prediction-optimization			X			X	X	
[50]	ROP optimization and reinforcement learning							X	
[56]	ROP prediction							X	
[9]	Anomaly detection						X	X	
[67]	Anomaly detection								X

2.4 Anomaly detection

The term "outlier" refers to a data point that could either be considered an abnormality or noise, whereas an "anomaly" refers to a special kind of outlier that is of interest to an analyst [3]. In the unsupervised scenario, where previous examples of interesting anomalies are not available, the noise represents the boundary between regular data and real anomalies – noise is often considered as a fragile form of outliers that does not always accomplished the criteria necessary for a data point to be considered unusual or anomalous enough [3]. It is the interest of the analyst that regulates the distinction between noise and an anomaly.

The study developed by Chandola et al. [20] has contributed to define the options for anomaly detection. In **statistical based models**, we found *parametric* and *no parametric* models. Gaussian, regression and mixture of distributions based models are part of *parametric techniques*. Besides, Histogram based and Kernel function based correspond to *non parametric techniques*. In **machine learning based model** the techniques are categorized as *classification*, *nearest-neighbor*, *clustering* and *spectral*. In classification group, we have neural networks, bayesian networks, support vector machine and rule based (decision tree). Nearest-neighbor methods involve distance and density based and clustering commonly known with k-means clustering and local outlier factor. While nearest-neighbor analyzes each instance with respect to its local neighborhood, clustering evaluates each instance with respect to the cluster it belongs. The last but not least, spectral techniques with principal component analysis. Two recently surveys [8] [45] have developed their taxonomy analysis based on the study developed by Chandola et al. [20]. More methods inside this classifications have been developed like isolation forest [41], autoencoders [28], and hierarchical temporary memory [4]. In addition, the notions of prediction and anomaly detection are intimately related [3]. Outliers are values that deviate from expected (or predicted) values on the basis of a particular model. Linear models focus on the use of inter-attribute dependencies to achieve this goal. In the classical statistics literature, this process is referred to as regression modeling.

The case of sensor network [20] contributes to address in our research of anomaly detection techniques. In this case, one or more sensors are faulty or they are detecting events, data is generated in a streaming mode, environment and the communication channel induce noise and missing values, the model requires to operate online and due to severe resource constraints the technique needs to be lightweight and distributed data mining approach. The techniques used in the sensor network corresponds to bayesian network, rule-based system with decision tree, Parametric statistical modeling, nearest neighbor-based and spectral techniques according to the survey [20].

The problem of detecting anomalies in streaming data has the following characteristics [60]. Firstly, the stream is infinite, so any off-line learning algorithms that attempt to store the entire stream for analysis will run out of memory space. Secondly, the stream contains mostly normal instances because anomalous data are rare and may not be available for training. In this case, any multi-class classifiers that require fully labeled data will not be suitable. Thirdly, streaming data often evolve over time. Thus, the model must adapt to different parts of the stream in order to maintain high detection accuracy. These relevant issues will be considered in our design.

Highlights

- Machine learning models have been implemented in drilling operation where *random forest* and *neural networks* have achieved high accuracy predicting *rate of penetration*. Stationary analysis, assessment per batches and prediction of specific intervals have been incorporated in these researches. The study developed by Soares et al. [58], Hegde et al. [30] and Shi et al. [56] offer a comprehensive analysis in this field.

- In the case of anomaly detection, different approaches have been developed for drilling operation without a conclusive structure. In addition, the issue has not been addressed in depth for drilling parameters in the literature. Chandola et al. [20] and Aggarwal [3] offer an overview about the possible techniques that the oil & gas industry could implement in their monitoring of parameters.

Chapter 3

Alternative solutions and selection

Which are the models that fit the context of our research? and, *which are the characteristics of the selected models?* We are going to develop this answer in Chapter 3. Firstly, the machine learning rate of penetration models that fit with the characteristics of the data set are explained. In the second section, the objective functions and the optimization algorithms to achieve optimum parameters are defined. The last section corresponds to the description of the anomaly detection methods.

3.1 Models for rate of penetration

Flexibility in model form allows machine learning algorithms to overcome physical-based ROP methods inability to segment the drilling parameter space. With no predefined equation, specific hyperparameters to each algorithm control model architecture. The machine learning models explored in this research are: support vector regressive, ensemble models (random forest and extreme gradient boost) and neural networks. There are two important factors that drive these successful applications: usage of effective (statistical) models that capture the complex data dependencies and scalable learning systems that learn the model of interest from large datasets [21]. However, the general disadvantage of machine learning models is that increased model complexity also create downsides in reduce interpretability and risk of overfitting.

3.1.1 Support vector regression

Support vector machine introduced by Vapnik [64] is a model employed for regression and forecasting. The model evaluates the regression based on kernel functions, which are able to convert the lower-dimensional input data to a higher dimensional space in an implicit manner.

How does the support vector machine model work? With a training data where X represents the space of input patterns. For instances, the rate of penetration with the corresponding depth. The purpose is to find a function $f(x)$ that has at most ε deviation from the actually obtained targets y_i (For instances, rate of penetration ROP_i) for all the training data, and at the same time is as flat as possible. We will accept errors less that ε, any deviation larger that this will not be accepted. The case of linear functions f take the following structure [57]:

$$f(x) = \langle w, x \rangle + b \, with \, w \in X, b \in \mathbb{R} \tag{3.1}$$

The problem can be represented as a convex optimization problem:

$$\begin{gathered} minimize\,\frac{1}{2}\left\|w\right\|^2 \\ subject\,to \begin{cases} y_i - \langle w, x_i\rangle - b \leq \varepsilon \\ \langle w, x_i\rangle + b - y_i \leq \varepsilon \end{cases} \end{gathered} \tag{3.2}$$

The function f approximates all pairs (x_i, y_i) with ε precision.

Knowing that we have a case of non-linearity, *how would be the structure of support vector machine?* In the case of non-linearity, the function is represented as follows:

$$f(x) = w\varphi(x) + b \tag{3.3}$$

where $\varphi(x)$ is the higher-dimensional feature space converted from the input vector x. The wights vector (w) and the threshold b can be estimated by minimizing the following regularized risk function [24].

$$R(C) = C\frac{1}{n}\sum_{i=1}^{n} L(d_i, y_i) + \frac{1}{2}\left\|w\right\|^2 \tag{3.4}$$

where C is the penalty parameter of the error, d_i is the desired value, n is the number of observations, $C\frac{1}{n}\sum_{i=1}^{n} L(d_i, y_i)$ is the empirical error, $\frac{1}{2}\left\|w\right\|^2$ is the regularization term. L_ε can be determined as following equation:

$$L_\varepsilon(d, y) = |d - y| - \varepsilon\,|d - y| \geq \varepsilon \tag{3.5}$$

where ε is the acceptance error. The Eq. 3.3 can be expressed in a explicit from by introducing Lagrange multipliers $(\alpha_i - \alpha_i^*)$.

$$f(x, \alpha_i, \alpha_i^*) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)k(x, x_i) + b \tag{3.6}$$

where $k(x, x_i)$ is the kernel function. For instance, the radial base function non-linear kernel function is:

$$k_{rbf}(x, x_i) = exp\left[\frac{-(x - x_i)^2}{2\sigma^2}\right] \tag{3.7}$$

where the kernel coefficient is represented as $\gamma = \frac{1}{2\sigma^2}$.

Consequently, to implement our model the next question that we need to address is: *What are the required parameters to adjust in support vector regression?* Following the previous description and the the models described in Python packages, we need to control the following hyperparameters:

- Kernel function included linear, polynomial and radial basis functions, and sigmoid.
- Epsilon (ε)
- Penalty parameter C of the error term.
- Kernel coefficient (γ)

3.1.2 Decision tree and random forest regression

Decision tree is proposed by Quinlan [51] where the idea is to construct a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Meanwhile, *random forest* method introduced by Breiman [14] is a compilation of decision trees and their results are aggregated into one final outcome.

This method builds a predictor ensemble with a set of decision trees that grow in randomly selected subspace of data [11]. Each tree is formed by first selecting at random, at each node, a small group of input coordinates (features or variable) to split on and, secondly, by calculating the best split based on these features in the training set. The convergence of the model depends only on the number of strong features and not on how many noise variables are present. The model is fast and easy to implement, produce highly accurate predictions and can handle a very large number of input variables without over-fitting.

Now that we have the definition, *How is the structure of Random forest?* Random forest is a collection of randomized base regression tress $\{r_n(x, \Theta_m, D_n, m \geq 1)\}$, where $\Theta_1, \Theta_2, ..., \Theta_n$ are outputs of a randomizing variable Θ. We have a training sample $D_n = \{(X_1, Y_1), ..., (X_n, Y_n)\}$. The goal is to estimate the regression function $r(x) = E[Y|X = x]$ using the data D_n. The aggregated regression estimate is presented as:

$$\bar{r}_n(X, D_n) = E_\Theta \left[r_n(X, \Theta, D_n) \right] \tag{3.8}$$

where E_Θ represent the expectation with respect to the random parameter, subject to X and the data set D_n.

At each node, a coordinate of $X = (X^{(1)}, ..., X^{(d)})$ is selected, with the $j - th$ feature having a probability $p_{nj} \in (0, 1)$ of being selected. Once the coordinate is selected, the split is at the midpoint of the chosen side.

Each randomized tree $r_n(X, \Theta)$ outputs the average over all Y_i for which the corresponding vectors X_i fall in the same cell of the random partition as X. Let $A_n(X, \Theta)$ be the rectangular cell of the random partition containing X.

$$r_n(X, \Theta) = \frac{\sum_{i=1}^{n} Y_i I_{[X_i \in A_n(X,\Theta)]}}{\sum_{i=1}^{n} I_{[X_i \in A_n(X,\Theta)]}} I_{E_n(X,\Theta)} \tag{3.9}$$

where I is and *indicator function* that represent association of an element in a subset A of X (e.g. $I_A : X \to 0, 1$). Finally, taking expectation with respect to the parameter Θ, the random forest regression estimate takes the form:

$$\bar{r}_n(X)) = E_\Theta \left[r_n(X, \Theta) \right] = E_\Theta \left[\frac{\sum_{i=1}^{n} Y_i I_{[X_i \in A_n(X,\Theta)]}}{\sum_{i=1}^{n} I_{[X_i \in A_n(X,\Theta)]}} I_{E_n(X,\Theta)} \right] \tag{3.10}$$

How could we describe this model in our context? In *decision tree*, data set is divided in two groups according to the criterion.

Measured data with rotation speed lower than 70 rev/min are gathered together and ROP average performs the outcome for the entire group. Next, data with rotation speed greater than 70 rev/min are clustered with the equal process. Similar process is obtained with the additional features. Combining high weight on bit parameter with high rotation speed, we will achieve sever hole cleaning issues and ROP slows down. Example retrieved from Soares et al. [58].

With the model described, *What are the required parameters to adjust in Random forest regression?* To answer this question, we need to analyze the following hyperparameters:

- Number of features to be consider in each split.
- Number of trees.
- Minimum samples to split.
- How deep trees should grow.

Now, we are able to implement random forest algorithm. However, *Is there any disadvantage of the model?* The simple decision trees are straight forward to interpret, but such interpretability is greatly diminished as random forest algorithm averages out multiple deep decision trees to improve predictive accuracy.

3.1.3 Extreme gradient boosting

How does extreme gradient boosting work? The algorithm submitted by Chen et al. [21] combines all the predictions of a set of "weak" learners for developing a "strong" learner through additive training strategies. During training phase, parallel calculation are executed. This method targets to prevent over-fitting and optimize computation capabilities. The impact of the system has been widely recognize in a number of machine learning and data mining challenges [21].

The algorithm looks encouraging according to the results in different competitions. *What is the framework of extreme gradient boosting?* Based on the model given in Fan et al. [24], we are going to develop this answer. The first learner is fitted to the whole space of input data, for tackling the deficiency of a weak learner a second model is then fitted to these residuals. Until the stopping criterion is met, the fitting process is repeated for a few times. By the sum of the prediction of each learner, the latest prediction of the model is obtained. The prediction at step t is defined in the following function:

$$f_i^t = \sum_{k=1}^{t} f_k(x_i) = f_i^{(t-1)} + f_t(x_i) \tag{3.11}$$

where $f_t(x_i)$ is the learner at step t, f_i^t and $f_i^{(t-1)}$ are the predictions at step t and $(t-1)$, and x_i is the input variable. The model defines the following expression to evaluate the "goodness" of the model from the original function.

$$Obj^{(t)} = \sum_{k=1}^{n} l(\bar{y}_i, y_i) + \sum_{k=1}^{t} \Omega(f_i) \tag{3.12}$$

where l is the loss function, n is the number of observations and Ω is the regularization term defined in the following expression:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \left\|\omega\right\|^2 \tag{3.13}$$

where ω is the vector of scores in the leaves, λ is the minimum loss needed to further partition the leaf node.

Again, we want to know *what are the parameters that we need to tune during our implementation?* Understanding the previous model and the structure defined in Python model, the hyperparameters are:

- Learning rate
- Number of estimators
- Maximum depth
- Minimum child weight
- Gamma
- Sub sample

where minimum child weight stop trying to separate once sample size in a node goes below a limit, gamma is the minimum loss reduction required to make a further partition on a leaf node of the tree, and sub sample represents a sub sample ratio of the training instances [21].

What are the advantages of extreme gradient boosting? One of the advantage recognized in the literature is that the system runs more than ten times faster than existing popular solutions on a single machine and scales to billions of examples in distributed or memory-limited settings [24].

3.1.4 Neural networks

To analyze neural networks technique, we are going to start describing the configuration of the model explained from Bishop [12]. Neural networks consists of successive layers of adaptive weights. We can find three different types on layers, input layer, one ore more hidden layers, and an output layer in which each layer includes a number of units. Single layer are based on a linear combination of the input variables which is transformed by a non-linear activation function. At the same time, we can have networks having successive layers of processing units, with connections running from every unit in one layer to every unit in the next layer. In feedforward neural networks, the direction of the information transmission is from input units through activation of the hidden units to the outputs.

The weights of the layers are updated in the training procedure by minimizing the objective function between the actual outputs of the network and the desire value. *Back propagation* is one of the algorithms for training a neural network. Through the information transferring on layers one by one, the predicted output is obtained. The training stop if the error between the formed and the expected outputs is sufficient. If the error is not acceptable in the past stage, the weights are substituted on the interconnections that go into the output layer. The weights are adapted for all interconnections that go into the hidden layer. Until the last layer of weights has been adjusted the process is continued.

The back-propagation algorithm could present poor performance employing the standard gradient descent method to adjust the weights. We could observe poor performance as a consequence of slow convergence speed and prone to being stuck in a local minimum. Therefore, We can find the following variants on different optimizing strategies: gradient descent with momentum, penalty term, and adaptive learning rate (AdaGrad, RMSProp, Adam) [28].

Now, we want to obtain details about the structure of the model. Therefore, we need to address the following question: *How the layers and units interact to get the output that we require in our regression problem?* We have a network with d inputs, M hidden units and c output units. The following network that we are going to analyze corresponds to a transformation of the input variables by two successive single-layer networks. The output of the jth hidden unit is obtained by first forming a weighted linear combination of the d input values, as follow:

$$a_j = \sum_{i=1}^{d} w_{ji}^{(1)} x_i \tag{3.14}$$

where $w_{ji}^{(1)}$ stand for a weight in the first layer, going from input i to hidden unit j. The activation of hidden unit j is then obtained by transforming the linear sum in Eq. 3.14 using an activation function $g(\cdot)$ to obtain:

$$z_j = g(a_j) \tag{3.15}$$

the outputs of the network are obtained using a second layer of processing elements. For each output unit k, we construct a linear combination of the outputs of the hidden units of the form:

$$a_j = \sum_{j=1}^{M} w_{kj}^{(2)} z_j \tag{3.16}$$

the activation of the kth output units is then obtained by transforming this linear combination using a non-linear activation function, to obtain

$$y_k = \tilde{g}(a_k) \tag{3.17}$$

the notation $\tilde{g}(\cdot)$ is used for the activation function of the output units to emphasize that this need not be the same function as used for the hidden units. Combining Eq. 3.14, Eq. 3.15, Eq. 3.16, and Eq. 3.17 we obtain an explicit expression for the complete function in the form:

$$y_k = \tilde{g}\left(\sum_{j=0}^{M} w_{kj}^{(2)} g\left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i\right)\right) \tag{3.18}$$

We can extend this class of networks by considering further successive transformations of the same general kind, corresponding to networks with extra layers of weights.

Given the data set, *how the previous network can learn a convenient mapping?* Learning will be based on the definition of an error function, which is then minimized with respect to the weights and biases in the network. We can evaluate the derivatives of the error function (*back-propagation*) with respect to the weights and these derivatives can then be used to find weight values which minimize the error function, by using gradient descent or another optimization function. The function error is given by:

$$E^n = \frac{1}{2}\sum_{k=1}^{c}(y_k - t_k)^2 \tag{3.19}$$

where y_k is the response of the output unit k, and t_k is the corresponding target, for a particular input pattern x^n.

The back-propagation procedure for evaluating the derivatives of the error E^n with respect to the weights is:

1. Apply an input vector x^n to the network and forward propagate through the network $a_j = \sum_i w_{ji} z_i$ and $z_j = g(a_j)$ to find the activation of all the hidden and outputs units.
2. Evaluate the δ_k (*errors*) for all the output units using $\delta_k \equiv \frac{\partial E^n}{\partial a_k} = g'(a_k)\frac{\partial E^n}{\partial y_k}$
3. Back-propagate the $\delta' s$ using $\delta_j = g'(a_j)\sum_j w_{kj}\delta_k$ to obtain δ_k for each hidden unit in the network.
4. Then, with $\frac{\partial E^n}{\partial w_{ji}} = \delta_j z_i$ we can evaluate the required derivatives.

The derivative of the total error E can then be obtained by repeating the above steps for each pattern in the training set. We can sum over all patterns:

$$\frac{\partial E}{\partial w_{ji}} = \sum_n \frac{\partial E^n}{\partial w_{ji}} \tag{3.20}$$

In the context of neural network training, we usually do not care about finding the exact minimum of a function, but only in reducing its value sufficiently to obtain good generalization error [28].

How could we translate the previous theory to our problem?. Features such as depth, weigth on bit, rotation speed, and flow rate are feeding into the input layer and then combined in hidden layers to produce the output, in our case rate of penetration. Discovering new features that represent relations between these inputs, the result for the unit in the hidden layer produces an alteration of the precursor data [58]. Units in the second hidden layer link the outputs of the previous units and arrange their own features to obtain the rate of penetration predicted in the output layer. For instance, using back-propagation, method we can obtain the weights for each unit connection.

As previous analysis, we require to control some hyperparameters to achieve the desired performance. *Which are the hyperparameters that we need to tune?.* Therefore, we need to consider the following variables:

- Number of hidden layers
- Number of units in each hidden layer
- Activation function
- I_2 regularization (α)

However, knowing that neural networks have achieved high performance, we have to be aware about some challenges of the technique. *What are these challenges found by researches in geo-sciences [52]?.* To begin with iterpretability, the field is still far from achieving self-explanatory models, and also far from causal discovery from observational data. Secondly, deep learning models can fit observations very well, but predictions may be physically inconsistent or implausible, owing to extrapolation or observational biases. Third, deep learning models are needed to cope with complex statistics, multiple outputs, different noise sources and high-dimensional spaces, but the exact cause-and-effect relations between variables are not clear in advance and need to be discovered. Finally, deep learning methods are needed to learn from few labelled examples while exploiting the wealth of information in related unlabelled observations.

3.1.5 Performance metrics

How are we going to analyze the performance of the selected models?. In this section, we are working with regression task where we want to know what is the difference between the predictions of our models and the field data. The most common metrics used in the literature are **mean squared error**, **mean absolute error** and **root mean squared error**. The following functions define each metric:

The average of squared differences between predicted and actual values is represented in the following way:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(ROP_{field,i} - ROP'_{Model,i}\right)^2 \tag{3.21}$$

where, n is the number of points in the test data-set.

Alternative, *mean absolute error* (MAE) defined as the average of absolute differences between predicted values $ROP'_{model,i}$ and the actual values $ROP_{field,i}$.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|ROP_{field,i} - ROP'_{model,i}\right| \tag{3.22}$$

Root mean squared error (RMSE) is a useful expression which has been used in different research to compare the performance of the models [58] [34] [30]. Here, we describe the metric:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(ROP_{field,i} - ROP'_{model,i}\right)^2} \tag{3.23}$$

In addition, Soares et al. [58] has identified that a disadvantage of RMSE is a relative measurement, incapable of quantify error significance without knowledge of the quantity mean. Once we are evaluating the results of different formations, each of this formation has a different mean. Therefore, the error is normalized over the mean of rate of penetration of the test set as follow.

$$Normalized\, RMSE = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(ROP_{field,i} - ROP'_{model,i}\right)^2}}{\frac{1}{n}\sum_{i=1}^{n} ROP_{field,i}} \tag{3.24}$$

3.2 Drilling physical models: Rate of penetration and Mechanical specific energy

As aforementioned in the introduction, control parameters that can be manipulated by the drilling engineer on the rig are weight on bit, drilling rotational speed, and drilling fluid (mud) flow rate [30][34][58]. Strength of the rock, geological properties, maximum pump power correspond to uncontrollable parameters which cannot be changed by engineers while drilling a well. Response parameters (the objectives) are those which change when control parameters are changed: rate of penetration, mechanical specific energy, downhole vibrations, and torque on bit.

What are the critical variables of real-time tracking? Penetration rate, *lateral vibrations*, *mechanical specific energy* and *unconfined compressive strength* have been variables supervised in real-time operation [5]. These variables have been used at the rig site to make operational decisions, and in post-drill analysis to redesign the procedure. Efforts are geared to vary drilling parameter to achieve minimum *mechanical specific energy* and obtaining optimum *penetration rate* as formation strength are reasonable uniform within the same formation interval. Increase in *mechanical specific energy* is an indication of inefficient drilling which may be caused by shock and vibration, high stickslip, bit balling and bit wear. Meanwhile, optimal weight on bit and rotary speed have been controlled parameters to optimize drilling rate.

Therefore, *what functions could we use to optimize the parameters?* Drilling control parameters are drilling parameters which can be changed to control the response (objective function). Algorithms are then used (on fitted drilling models) to determine the best control parameters to implement ahead of the bit [30]. Model evaluation is performed by running simulations on data measured during drilling a well. *Rate of penetration* and *mechanical specific energy* are modeled using a physical-driven approach.

Based on the research developed by Soares et al. [58] and Hegde et al. [30], we are going to work with the model for *rate of penetration* proposed by modified *Bourgoyne and Young (1986)* and the model developed by *Teale (1965)* for *mechanical specific energy*. The evaluation of the best physical model for this variables is not part of our scope, that is the reason that we select the models with the best performance in the mentioned researches.

$$ROP = a_1 D^{a_2} WOB^{a_5} RPM_{a_6} q^{a_8} \tag{3.25}$$

where the function depends on the diameter of bit (D), weight on bit (WOB), rotational speed (RPM) and the flow rate (q). The model performance highly on coefficient bounds a_1, a_2, a_5, a_6, a_8.

In the case of *mechanical specific energy (MSE)*, we have:

$$MSE = \frac{WOB}{A_b} + \frac{120\pi\, RPM\, TOB}{A_b\, ROP} \tag{3.26}$$

where the function depends on the cross sectional area of bit (A_b), weight on bit (WOB), rotational speed (RPM) and torque on bit (TOB).

3.3 Models for misbehavior detection

In this section, we must answer key questions to define among several techniques (See Chapter 2), which one will fit in our research. According to Chandola et al. [20], to define a method *"the problem is determined by several different factors such as the nature of the input data, the availability or unavailability of labels as well as the constraints and requirements induced by the application domain"*. Therefore, before the definition of alternatives we need to answer the following question: *what are the characteristics of our data set in the context of anomaly detection?*. The structure proposed by Chandola et al. [20] allows us to characterize the data based on nature of input data, type of anomaly, labels, and output. The respective analysis is presented in the following items:

(a) **Nature of input data:**

- *Multivariate* (Multiple attributes): Weight on bit, rotational speed, drilling fluid rate, temperature, surface torque are some of the variables that we obtain on real time operation.
- *Sequence of data:* Data instances are linearly ordered. The drilling rig continuously receive information *equidistant* where the values of temporal attribute are equally spaced by a time step Δ. At the same time, we have valuable information in each depth of the well consider as a sequence as well.

(b) **Type of anomaly:** Anomalies can be classified into *point anomalies*, *contextual anomalies* and *collective anomalies.*

- *Point Anomalies:* If an individual data instance can be considered as anomalous with respect to the rest of data, then the instance is termed a point anomaly [20]. We could have scenarios where *rate of penetration* have extreme variations as a consequence of high or low rotational speed, then we can consider this point as an anomaly.
- *Contextual anomaly*: If a data instance is anomalous in a specific context, but not otherwise, then it is termed a contextual anomaly [20]. For instance, the longitude and latitude of a location are the contextual attributes in spatial data sets. In our case, we could have the depth as a contextual attribute that determines the position of an instance on the entire sequence. In addition, the top of the formations could delimit the specific context in our study. In means, that some variables could have

specific values in a defined formation but this values could have a relevant change in the following formation. Even though, it is not considered as an anomaly.

- *Collective anomaly.* The individual data instances may not be anomalies by themselves, but their occurrence together as a collection is anomalous [20]. This scenario have relevance in our study where small variations in a sequence of the same variable could generate non productive time during the operation. According to Chandola et al. [20], collective anomalies can occur only in data sets in which data instances are related. Here, we have interaction of multiple variables in each depth instance.

(c) **Data labels:** Anomaly detection techniques can operate in one of the following three modes: *supervised*, *semi-supervised* or *unsupervised* anomaly detection based on the availability of labels.

- *Supervised anomaly detection* techniques assume the availability of a training data set that has labeled instances for normal as well as anomaly classes. However, this case does not apply in our data set where we are collecting streaming real time data without any label.
- In the scenario of *semi-supervised anomaly detection* techniques assumes that the training data has labeled instances only for the normal class. However, in our case normal class is not label either. One approach is to build a model for the class corresponding to normal behavior, and use the model to identify anomalies in the test data.
- *Unsupervised anomaly detection*, techniques do not require training data and make the assumption that normal instances are far more frequent than anomalies in the test data. According to Chandola et al. [20], semi-supervised techniques can be adapted to operate in an unsupervised mode by using a sample of the unlabeled data set as training data. This last scenario could apply in our case and base of distances or density or another techniques we could determine the anomalies.

(d) **Output of anomaly detection:** The two type of outputs that we can produce are *scores* or *labels.*

- *Scores:* The output is a ranked list of anomalies. Depending on the degree to which that instance is considered an anomaly this technique assign an anomaly score to each instance in the test data. Our data set is not static, consequently we do not have the full data set to give a rank based on critically.
- *Labels:* This category assigns a label *normal* or *anomalous* to each test instance. As aforementioned, we have streaming data where we could label the respective anomalies being a feasible alternative.

Therefore, we have a multivariate data set which is captured as a sequence (streaming data) where we could find point, contextual and collective anomalies, in an scenario of unsupervised learning obtaining normal or anomalous labels. According to this scenario, *What would be the ideal technique that we need to implement to identify these anomalies?*. To give an answer about the ideal technique, Ahmad et al. [4] have defined the ideal characteristics of a real-world anomaly detection summed up as follows:

1. Before receiving the subsequent x_{t+1} state, the algorithm must identify x_t state as normal or anomalous. It means that predictions must be made online.
2. Without a requirement to store the entire stream, the algorithm must learn continuously.
3. The method must run without data labels or manual parameter tuning. It means that the algorithm is an unsupervised automated fashion.
4. Algorithms must adapt to dynamic environment. Data stream is often non-stationary.
5. The detection should be as soon as possible.
6. False positive and false negatives should be minimize by the algorithm.

This research has examined the following techniques: k-means clustering, decision trees (isolation forest), one class support vector, and deep neural networks (autoencoder).

3.3.1 k-means clustering

A clustering algorithm can be employed to reveal the groups in which feature vectors are clustered in the l-dimensional feature space [62]. Clusters are formed from a series of examples, any example that is found not to match an existing cluster would be defined as an anomaly [2]. Clusters are described as continuous regions of this space containing a relatively high density of points, separated from other high density regions of relatively low density of points [62].

"The purpose of clustering methods is to group patterns on the basis of a similarity (or dissimilarity) criteria where groups (or clusters) are set of similar patterns" [25]. For metric representation, the most popular dissimilarity measure for metric representations is the *euclidean distance* [25]. We can find two categories in clustering techniques: *hierarchical* and *partitioning*. In this research we will focus in *partitioning* technique where we obtain separations hypersurfaces among clusters.

What are the main steps to develop a clustering task? According to Theodoridis [62], the main steps are proximity measure, clustering criterion and clustering algorithm. Proximity measure quantify how "similar" or "dissimilar" two features vectors are. The clustering criterion may be expressed via a cost function. Next step is to chose a specific algorithm scheme that resolve the clustering structure.

Now that we have the steps, we need to understand the structure of the technique. Therefore, we will answer the following question: *What are the functions and metrics determined to create the clusters in our dataset?* Let $X = \{x_1, ..., x_n\}$ be a data set composed by n patterns. The *set of centroids* V is defined as the set $V = \{v_1, ..., v_c\}$ where each element v_i is called centroid. The *Voronoi region* R_i of the centroid v_i is the set of centroids for which v_i is the nearest vector:

$$R_i = \left\{ x \in X | i = arg\,\min_j \|x - v_j\|^2 \right\} \tag{3.27}$$

Starting from the finite data set X, the method moves iterative the k centroids to the arithmetic mean of their Voronoi set $R_{i,i=1,\ldots,k}$. A condition for a set of centroids V to minimize the *empirical quantization error*:

$$E(X) = \frac{1}{2n} \sum_{i=1}^{k} \sum_{x \in R_i} \|x - v_i\|^2 \tag{3.28}$$

each centroid v_i must fulfill the *centroid condition*. A finite data set X and with *euclidean distance*, the centroid condition is defined as follow:

$$v_i = \frac{1}{|R_i|} \sum_{x \in R_i} x \tag{3.29}$$

Then, the following steps represent the *batch k-means* algorithm:

1. Choose the number k of clusters.
2. Initialize the *set of centroids* V with vectors randomly selected from X.
3. Compute the *Voronoi* set R_i associated to the centroid v_i.
4. Move each centroid to the mean of its *Voronoi* set using Eq. 3.29.
5. Return to step 3 if any centroid has changed otherwise return the set of centroids.

In general, the distance of a data point to its k-nearest neighbor (or other variant) is used in order to define proximity. Data points with large k-nearest neighbor distances are defined as outliers. Distance-based algorithms typically perform the analysis at a much more detailed granularity than the other two methods [3] On the other hand, this greater granularity often comes at a significant computational cost.

Based on the previous description, *What are the parameters that we need to tune?* The following parameters are required in our implementation:

- The number of clusters to form as well as the number of centroids to generate.
- Maximum number of iterations of the k-means algorithm for a single run.

K-means clustering has been widely used in static data. *How are we going to performance with real time classification?* Three approaches can be taken to perform real-time classification [2]: (i) to shorten the computational search time needed for classifying the pattern, the size of an input can be reduced; (ii) to limit the computation time to search the complete space, the maximum number of clusters can be restricted; and (iii) the number of clusters can be reduced from the complete set of existing clusters to some subset of the complete search space tested in real time.

3.3.2 Isolation forest

Isolation forest proposed by Liu and Zhou [41] builds an ensemble of trees where the instances with short average path length on the trees are consider anomalies. The algorithm sorts data points according to their average path lengths and anomalies are points ranked at the top of the list. It is also noteworthy that isolation tree algorithm are virtually parameter-free; this makes their applicability particularly simple [3]. This alternative detects anomalies based on the mechanism of isolation without any distance or density measure. This condition eliminates a major computational cost. Moreover. it considers that the data set has *few* anomaly instances and the anomalies are very *different* from those of normal instances. Therefore, *how does the method identify the isolated point in our data set?* The process is described in the following items:

- The training stage shapes isolation trees using sub-samples of the giving training set. Isolation trees are built by recursively partitioning a subsample X' until all instances are isolated. Multiple times the partition is done independently. The algorithm contains two input parameter, the subsampling size ψ that control the training data size, and the number of trees t controls the ensemble size.

- Then, test instances passes through isolation trees to obtain an anomaly score for each instance. A single path length $h(x)$ is derived by counting the number of edges e from the root node to an external node as instance x traverses through an isolation tree. Once the traversal reaches a predefined height limit h_{lim}, the return value is e.
- When $h(x)$ is obtained for each tree of the ensemble, anomaly score is computed.

As previous analysis, we are interested to identify the parameters to obtain good performance of the model. According to the model, we have:

- Number of trees to build and subsampling size.
- The tree height limit corresponds to the evaluation parameter.

What are the advantage of this algorithm? According to Liu and Zhou [41]:

- Isolation forest fit in high dimensional problems with a large number of irrelevant attributes and when anomalies are not available in training sample.
- The model exploit sub-sampling technique to obtain a low linear time-complexity and a small memory-requirement.
- The method converges quickly with very small number of trees.
- Lower computational costs is obtained because not distance or density measure is required to be defined and calculated.

3.3.3 Autoencoder

Autoencoders train to copy their input to their output [28]. The neural network has a hidden layer h that define a code used to describe the input. This network has two parts, an encoder function $h = f(x)$ and a decoder that obtain a reconstruction $r = g(h)$.

How does the autoencoder algorithm obtain a reconstruction error? In the training phase, we have the training set $x(1), x(2), ..., x(m)$. Each data sample $x(i)$ is represented by a vector of D different variables. The data is compressed into lower dimensional subspace and reproduce the output $\{\hat{x}(1), \hat{x}(2), ..., \hat{x}(m)\}$. Therefore, the reconstruction error used as anomaly score is defined as follow [53]:

$$Err(i) = \sqrt{\sum_{j=1}^{D}(x_j(i) - \hat{x}_j(i))^2} \tag{3.30}$$

In the test step, the test data is projected into the subspace and reconstruct the original data. For normal instances, the reconstruction error has low values that satisfy the normal correlation learned during the test phase. On the other hand, reconstruction error has high value with anomalous samples.

Activation of unit i in layer l is described in the following equation:

$$a_i^{(l)} = f\left(\sum_{j=1}^{n} W_{ij}^{(l-1)} a_j^{(l-1)} + b_i^{(1)}\right) \tag{3.31}$$

where W and b represent the weight and bias parameters. The input layer $a^{(1)} = x$, the last layer $a^{(3)} = \hat{x}$. The goal is to minimize the following objective function Eq. 3.32 with respect to W and b:

$$J(W, b) = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|x(i) - \hat{x}(i)\right\|^2\right) + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(W_{ji}^{(l)}\right)^2 \tag{3.32}$$

where λ expresses the strength of regularization, n_l represent the number of layers in the network and s_l determines the number of units in layer L_l.

Neural networks are highly dependent of hyperparameters. *What are the main hyperparameters required to tune our algorithm?*

- Optimizer
- Loss function
- Learning rate
- Epochs
- Batch size
- Number of units per layer
- Number of layers

3.3.4 One-class support vector machine

This model is porposed by Schölkopf et al.[55]. The algorithm is designed specifically to detect outliers. The idea is separate all the possible faulty samples from normal samples [42]. Given the normal operating regime, this technique compute the margin support or the boundary that accommodates most of the training points. If the test sample falls within the boundary y, it is categorized as normal operating data, otherwise it is classified as an outlier or a faulty data point [42]. Therefore, the objective of the algorithm is to develop a classifier or hyperplane in the feature space which returns a positive value for all samples that fall inside the normal cluster and a negative value for all values outside this cluster and maximize the perpendicular distance of this hyperplane from the origin.

How is structured the optimization function in one-class support vector machine? The optimization problem is described as follow:

$$\begin{gathered} min_{w,\xi_i,b}\tfrac{1}{2}\left\|w\right\|^2 + \tfrac{1}{\nu m}\textstyle\sum_{i=1}^{m}\xi_i + b \\ subject\,to\,w\cdot\phi(x_i) + b + \xi_i \geq 0,\, \xi_i \geq 0, i = 1...m \end{gathered} \tag{3.33}$$

where Lagrange multipliers $\alpha_i \geq 0$ and $\eta_i \geq 0$ are introduced. The partial derivatives of the Lagrangian are set to zero as follows:

$$L(w,\xi,b) = \frac{1}{2}\left\|w\right\|^2 + \frac{1}{\nu m}\sum_{i=1}^{m}\xi_i + b - \sum_{i=1}^{m}\alpha_i(w\cdot\phi(x_i) + b + \xi_i) - \sum_{i=1}^{m}\eta_i\xi_i \tag{3.34}$$

$$\frac{\partial L}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{m}\alpha_i\phi(x_i) \tag{3.35}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \rightarrow \alpha_i = \frac{1}{\nu m} - \eta_i,\, \eta_i \in (0,1) \tag{3.36}$$

$$\frac{\partial L}{\partial b} = 0 \rightarrow \sum_{i=1}^{m}\alpha_i = 1 \tag{3.37}$$

Here, the variable ν is the maximum value of the fraction of training data set errors, ξ_i is the slack variable to relax the optimality constraints.

Therefore, *What are the main parameters for one-class support vector?*

- Kernel (linear, poly, radial basis function, sigmoid)
- Degree (in case that we use kernel poly)

- Gamma (γ)
- Nu (ν)

Highlights:

- The machine learning models selected are defined with their respective optimization function to predict and detect anomalies. The identification of parameters is fundamental to implement and optimize. Relevant parameters allow us to find the best performance for each algorithm.
- The metrics defined for our analysis include *mean squared error*, *root mean squared error* and *normalized mean squared error*.
- The physical model of *rate of penetration* and *mechanical specific energy* give us an insight about the main parameters that has been studied and considered in drilling operation which include *weight on bit*, *rotary speed*, *torque on bit*, *flow* and *diameter of the hole*.
- In addition, in this section was described the characteristics of our data set obtained in drilling operation based on the categories established by Chandola et al. [20]. In case that you are interested to find insight of the anomaly detection methods, this reference contributes to clarify in this field.

Chapter 4

Implementation

4.1 Implementation for rate of penetration prediction

Drilling rigs are constantly receiving drilling data in the cabin of the driller. It means, that constantly we have incremental training data accessible. Machine learning models depends on the size of data available. Some model have good performance with small proportion of data available for training. However, methods as neural networks works better once more data is available. With the following implementation, we are going to analyze this scenario given answer to the following question: *Could machine learning models learn more about drilling parameters with incremental data?*.

The first exercises is developed splitting the data set in nine (9) batches increasing the data available per batch. In this scenario we are simulating real-time operation with incremental data to train. The design is presented in Figure 4.1. This architecture will be used in (i) full data set per batches and (ii) analysis per formation.

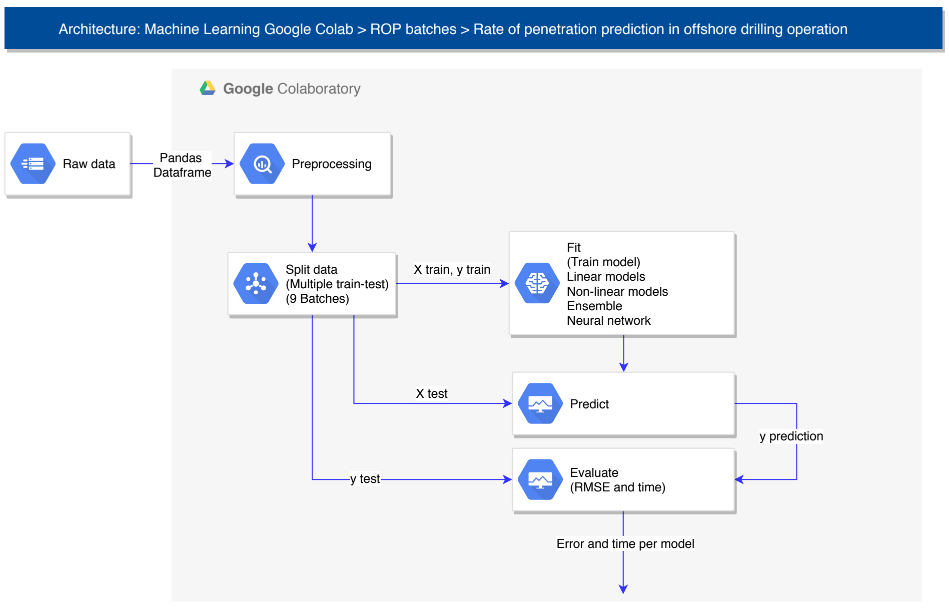

Architecture: Machine Learning Google Colab > ROP batches > Rate of penetration prediction in offshore drilling operation
Google Colaboratory
Raw data
Pandas Dataframe
Preprocessing
Split data (Multiple train-test) (9 Batches)
X train, y train
Fit (Train model) Linear models Non-linear models Ensemble Neural network
X test
Predict
y prediction
y test
Evaluate (RMSE and time)
Error and time per model

FIGURE 4.1: Prediction of rate of penetration per batches

The analysis include the evaluation of linear model, non-linear model, ensemble and neural network. The metric defined to analyze the performance of the model is *Root mean squared error* and *Normalized root mean squared error*. The last indicator allow us to compare the error between formations. As aforementioned in Chapter 1 and 2, the process follows the main steps defined by Theodoridis et al [62].

The data set contains many variables. *What are the most importance features?* Based on *Gini Importance* method measuring *Mean Decrease in Impurity (MDI)* we are going to determine the parameters that have more impact in the prediction of rate of penetration. In Figure 4.2, we defined the steps required. At the same time, we compared the performance (RMSE and time) of the prediction using all variables and only the parameters that capture the 95% of importance. The result of this analysis will allow to identify if decreasing the number of features, we could gain in accuracy and run time or we should continue with all the feature. Normally, the more data we have the more the performance. However, more data could bring more noise.

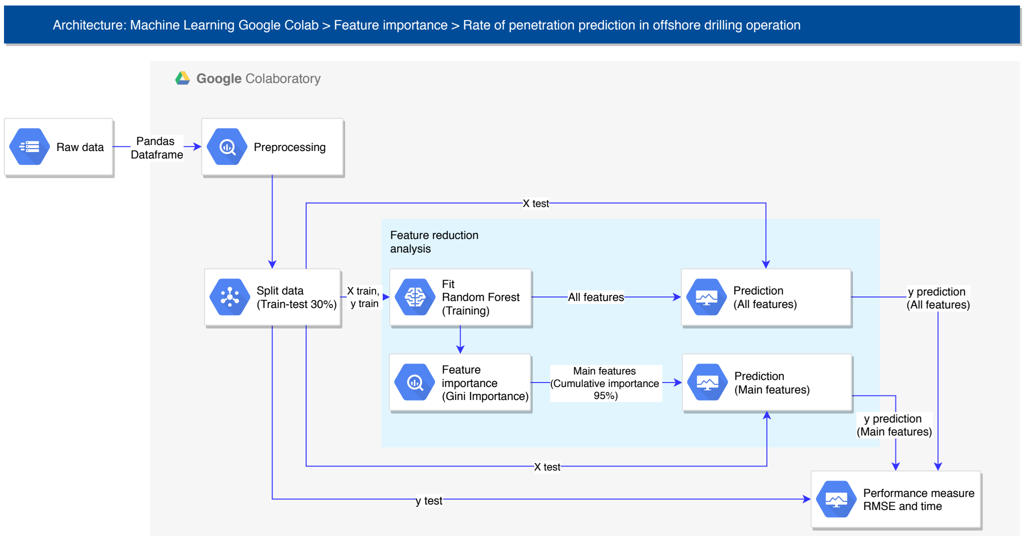

Architecture: Machine Learning Google Colab > Feature importance > Rate of penetration prediction in offshore drilling operation
Google Colaboratory
Raw data
Pandas Dataframe
Preprocessing
X test
Feature reduction analysis
Split data (Train-test 30%)
X train, y train
Fit Random Forest (Training)
All features
Prediction (All features)
y prediction (All features)
Feature importance (Gini Importance)
Main features (Cumulative importance 95%)
Prediction (Main features)
y prediction (Main features)
X test
y test
Performance measure RMSE and time

FIGURE 4.2: Implementation for feature importance

In addition, as we mention in Chapter 3, machine learning parameters require to be tune. To optimize the hyperparameters we implement *cross-validation* stage. The architecture to describe the process is presented in Figure 4.3. Assessing the machine learning models, we defined two blocks depending on the algorithm. For instance, in the case of *random forest* and *xgboost*, normalization is not required while *support vector regression* and *neural networks* are affected in case of unbalanced data where one variable dominate and the rest of the variables will be almost ignored. Posterior to the prediction, the data is re-scaled to improve the interpretation and visualization of the results.

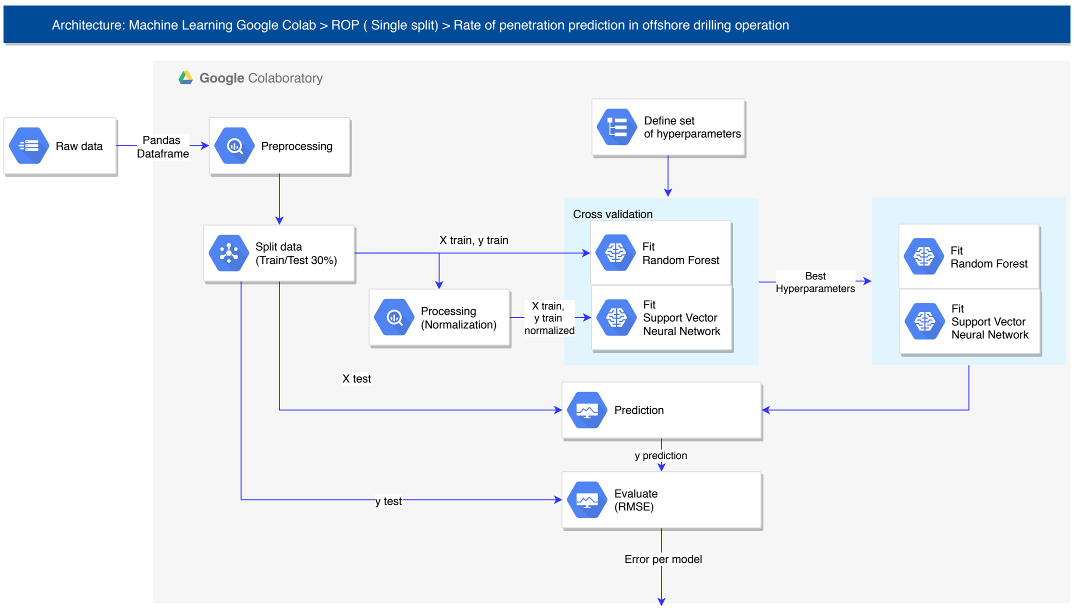

Architecture: Machine Learning Google Colab > ROP (Single split) > Rate of penetration prediction in offshore drilling operation
Google Colaboratory
Raw data
Pandas Dataframe
Preprocessing
Define set of hyperparameters
Cross validation
Split data (Train/Test 30%)
X train, y train
Fit Random Forest
Best Hyperparameters
Fit Random Forest
Processing (Normalization)
X train, y train normalized
Fit Support Vector Neural Network
Fit Support Vector Neural Network
X test
Prediction
y prediction
y test
Evaluate (RMSE)
Error per model

FIGURE 4.3: Rate of penetration prediction in stationary analysis and optimization of hyperparameters

Implementing *cross-validation* involves randomly *k-fold CV* dividing the set of observations into k groups, or folds, of approximately equal size. Figure 4.4 represents the distributions of the folds in each split which contributes to fin the best parameters for the machine learning model. The first fold is treated as a validation set, and the method is fit on the remaining $k-1$ folds [38]. The mean squared error is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set.

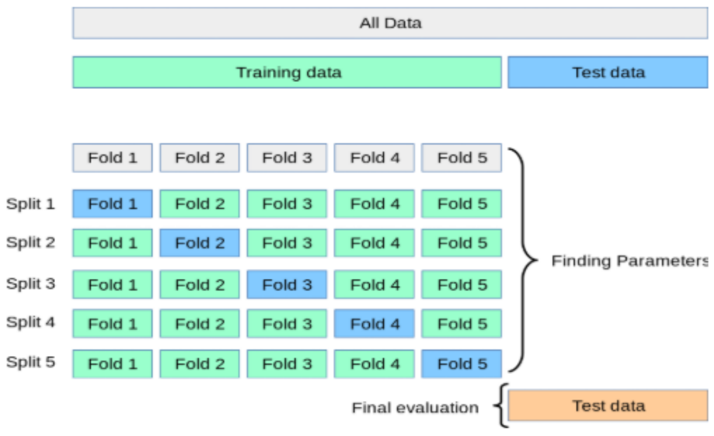


FIGURE 4.4: Cross validation architecture [47]

4.2 Implementation for anomaly detection

As aforementioned in the introduction (Chapter 1), it is difficult for humans to recognize abnormal state and normal state using raw data [53]. Therefore, real-time monitoring centers require techniques to flag in case of anomaly events. We are going to explore machine learning models to contribute in the drilling operation optimization. The analysis is divided in two categories: *stationary analysis* and *time series analysis*.

(a) In the case of stationary analysis, we are going to assess *k-means clustering*, *one-class support vector machine*, *isolation forest*, and *autoencoders* which are structured in the following diagrams (Figures 4.5, 4.6, 4.7, and 4.8).

(b) For time series analysis, we are going to design an architecture based on regression models (*random forest*, *xgboost*, *k-nearest neighbor*, *decision tree*, *support vector regression*, and *neural network*), and difference between prediction and measured data (Figure 4.9.

Before to start, some critical observations that arise in the context of outlier detection are as follows [3]:

- Data normalization is important to avoid imbalance losing importance of some variables.
- We need to differentiate between noise and interesting anomalies. Most application domains contain noisy or incomplete data or data that has errors. For example, sensor data often contains noise because of defects in transmission or failure in the data-collection hardware.
- Data cleaning is itself a key application of outlier analysis.
- Exploratory and visual analysis can be helpful at all stages of outlier analysis.
- A human in the loop can more easily generate labels in conjunction with unsupervised outlier analysis algorithms. Unsupervised and supervised algorithms can be used in an iterative way in conjunction with a human in the loop in order to generate labels. Even a small amount of labeled data can significantly improve the effectiveness of outlier analysis algorithms.
- Outlier ensembles can be used to reduce risk. By combining a small number of models that are known to work well across a wide variety of data sets, it is possible to consistently obtain results of good quality.

These observations will contribute with the design of our architecture for stationary analysis and time series analysis.

4.2.1 Stationary anomaly detection

The pipeline for **k-means clustering** is presented in Figure 4.5. The data is normalized to avoid the impact of imbalance and the two (2) principal component is calculated. The next stage, *k-means clustering* method is fitted. In this case, all the data is injected to the model to obtain the clusters. For this stage, we define an initial number of clusters k. *Elbow curve analysis* allows us to identify the correct number of clusters based on the variance score. It means, that we can obtain the number of cluster with the lowest error. Considering the distance of each point and the centroids, we are able to determine the anomalies. Those points considered far from the centroids will be determined as misbehavior according to the threshold for outlier fraction.

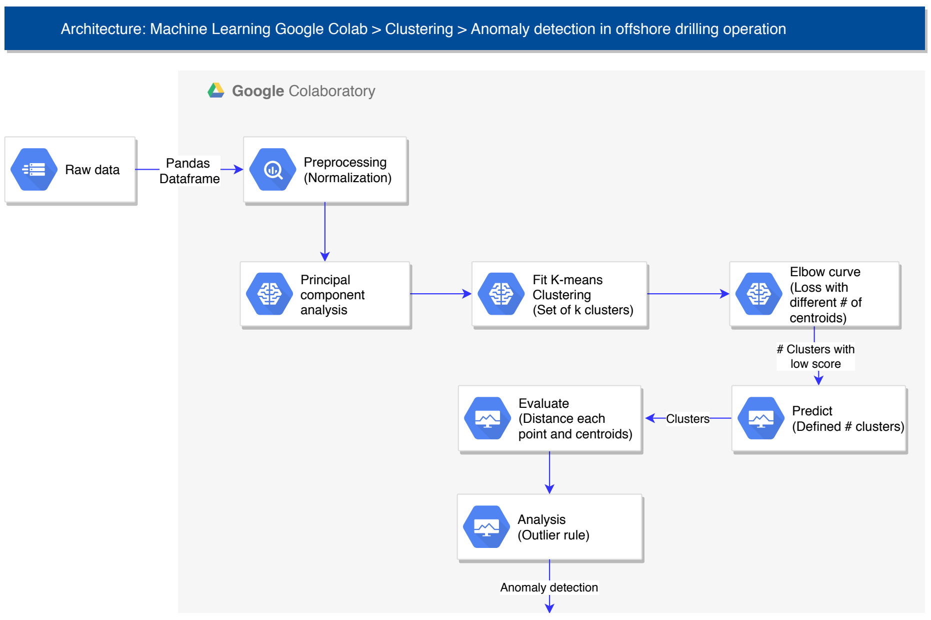

Architecture: Machine Learning Google Colab > Clustering > Anomaly detection in offshore drilling operation
Google Colaboratory
Raw data
Pandas Dataframe
Preprocessing (Normalization)
Principal component analysis
Fit K-means Clustering (Set of k clusters)
Elbow curve (Loss with different # of centroids)
# Clusters with low score
Predict (Defined # clusters)
Clusters
Evaluate (Distance each point and centroids)
Analysis (Outlier rule)
Anomaly detection

FIGURE 4.5: Anomaly detection based on *k-means clustering* technique.

In the case of **isolation forest**, the model identifies the short path in the ensemble which is labeled as an anomaly as was mentioned in Chapter 3. The basic version of the isolation tree, when grown to full height, is parameter-free. Unlike traditional histograms, *isolation forests* are not parameterized by grid-width and are therefore more flexible in handling data distributions of varying density. This characteristic is always a significant advantage in unsupervised problems like outlier detection. This methodology is straightforward, where additional steps are not required to identify misbehavior and all the data is injected to the model. The process is presented in Figure 4.6. This model does not require normalized data. As the previous algorithm, outlier fraction must be defined. It means, that we need to define the proportion of points that could be consider as an outlier. Probably we need to analyze multiple data set from drilling data to define this threshold.

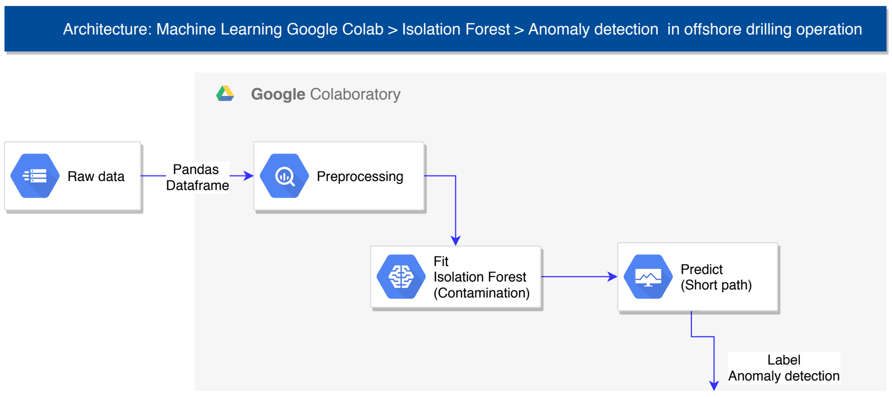

Architecture: Machine Learning Google Colab > Isolation Forest > Anomaly detection in offshore drilling operation
Google Colaboratory
Raw data
Pandas Dataframe
Preprocessing
Fit Isolation Forest (Contamination)
Predict (Short path)
Label Anomaly detection

FIGURE 4.6: Anomaly detection based on *isolation forest* technique.

In the case of **one-class support vector** (Figure 4.7), we normalized the data. Then we injected all the data set in the model. We trained to infers the properties of normal cases and from these properties can predict which examples are unlike the normal example. Once the model identify the abnormal points, it is labeled with 1 as anomaly and 0 as normal.

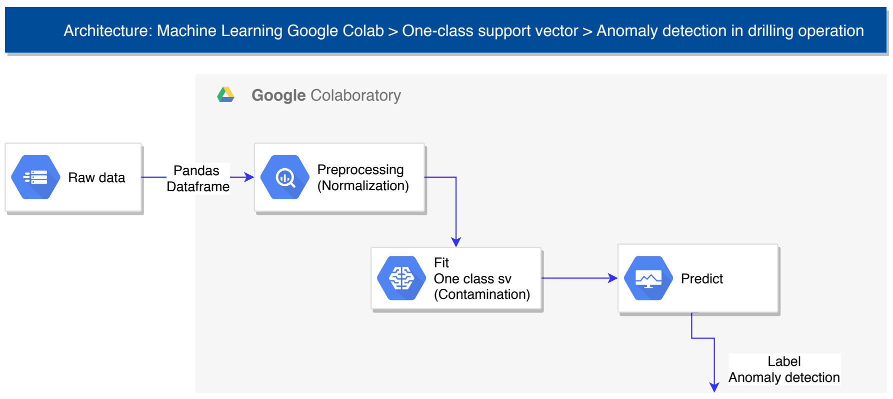

Architecture: Machine Learning Google Colab > One-class support vector > Anomaly detection in drilling operation
Google Colaboratory
Raw data
Pandas Dataframe
Preprocessing (Normalization)
Fit One class sv (Contamination)
Predict
Label Anomaly detection

FIGURE 4.7: Anomaly detection based on *one-class support vector* architecture

The implementation of **autoencoder** is presented in Figure 4.8. This pipeline require normalization of data. We split the data where 80% correspond to training data and 20% to test data. The model stack is defined with the section of decoder layers and the respective number of units as well as the encoder structure. For this neural network we define optimizer, loss function, epoch, batch size, and validation split. The model measure the reconstruction error metric to define the anomalies. At the same time, we can identify the cause-root of the anomaly. For instance, we could diagnose if *weight on bit*, *rotation speed* or *flow* are affecting the *rate of penetration*.

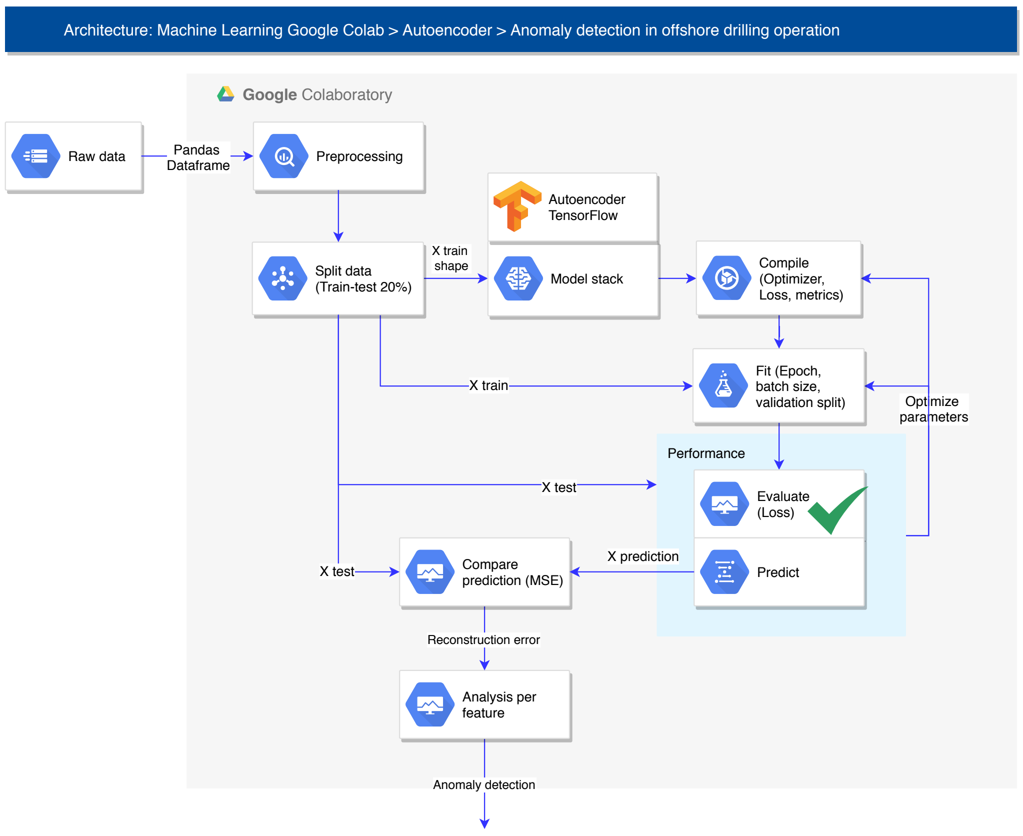


FIGURE 4.8: Anomaly detection based on *autoencoder* technique.

The architecture of the neural network is presented in Table 4.1. One should select the complexity of the neural network depending on the amount of available training data [3]. In our case, we are exploring the algorithm. Further research will require optimization of parameters.

TABLE 4.1: *Autoencoder* architecture

Layer	**Output shape**	**Param #**
Input 1	16	0
Dense 1	12	204
Dense 2	6	78
Dense 3	3	21
Dense 4	6	24
Dense 5	12	84
Dense 6	16	208

4.2.2 Time series anomaly detection

In the case of time series anomaly detection, we have similar configuration as the architecture developed for rate of penetration prediction. In this case we are assessing *random forest*, *xgboost*, *k-nearest neighbor*, *decision tree*, *support vector regression*, and *neural network* to predict the next segment of points. Once we have the prediction, we can calculate the difference between the prediction and the measured data. Defining a threshold of outlier fraction we can control the definition of anomaly. At the end, base on the conditional function we classify our data as normal or anomaly which is represented in Figure 4.9.

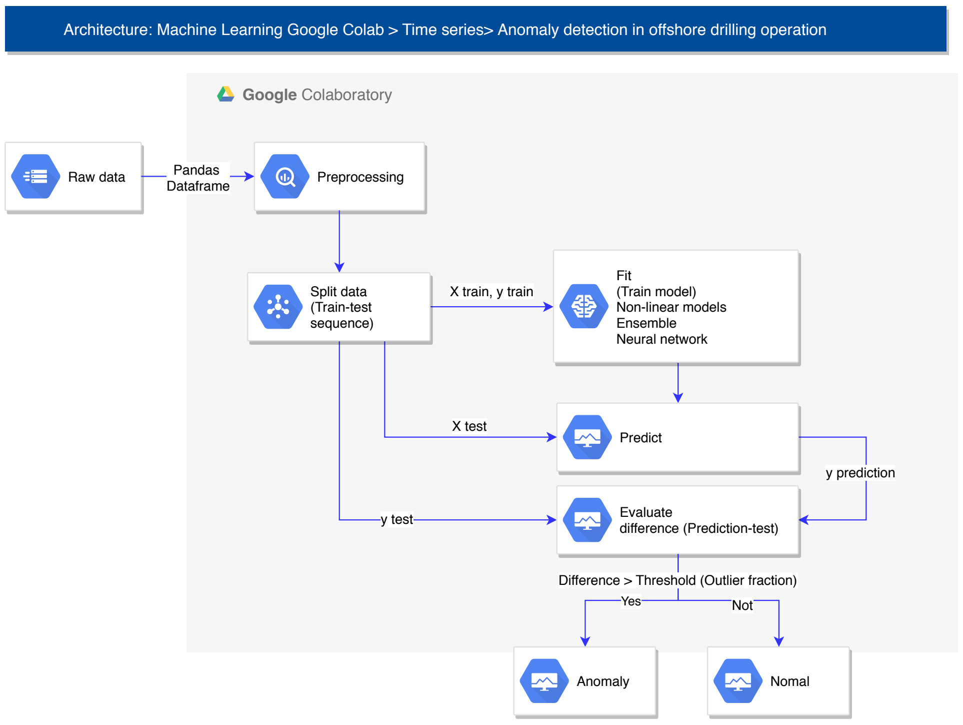

Architecture: Machine Learning Google Colab > Time series> Anomaly detection in offshore drilling operation
Google Colaboratory
Raw data
Pandas Dataframe
Preprocessing
Split data (Train-test sequence)
X train, y train
Fit (Train model) Non-linear models Ensemble Neural network
X test
Predict
y prediction
y test
Evaluate difference (Prediction-test)
Difference > Threshold (Outlier fraction)
Yes
Not
Anomaly
Nomal

FIGURE 4.9: Anomaly detection based on *time series* architecture

This structure is highly dependent on the accuracy of our prediction. However, machine learning algorithms have high performance in non linear behaviors.

Highlights:

- The architectures designed for *rate of penetration prediction* cover *stationary analysis* injecting all the data set selecting test set randomly. This include (i) feature importance, (ii) reduction of features and (iii) hyperparameter optimization analysis. In addition, *predictions per batches* is structured to evaluate (i) all data set and (ii) formations simulating streaming data. Linear, non-linear, ensemble and neural networks are incorporated in this project.
- For anomaly detection, the techniques are classified for *stationary analysis* and *time series detection*. First alternative will contribute for post analysis of drilling engineer where all the data is injected to detect misbehaviors with *isolation forest*, *k-means clustering*, *one-class support vector* and *autoencoders*. Meanwhile, time series detection is proposed based on regression models (linear, non-linear, ensemble and neural networks) to predict the next interval and obtain the difference respect to measured data.

Chapter 5

Experimental results and evaluation

5.1 Data set of drilling operation

The data set selected for this project corresponds to drilling data from Forge project Well 58-32 Milford, Utah (USA) [46] and Well 21-31 Fallon, Nevada (USA) [54] as was mentioned in Chapter 1 (See location in Figure 5.1).

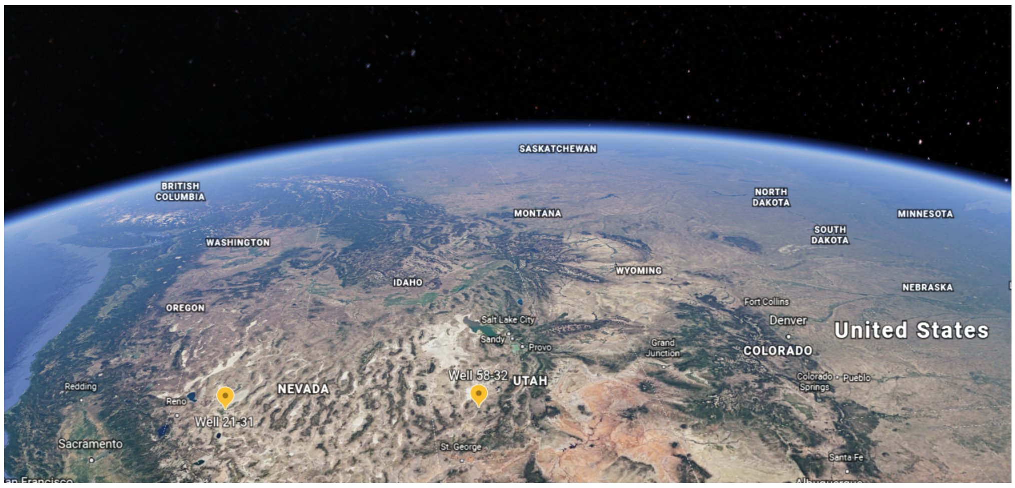


FIGURE 5.1: Location of Well 58-32 Milford, Utah (USA) and Well 21-31 Fallon, Nevada (USA)

About **Well 58-32 Milford, Utah (USA)** [10]:

(a) Well 58-32 was drilled vertically in 58 days (July-August 2017) to a depth of 7,536 ft.

(b) The intent of the drilling was to determine the characteristics of the rock within the target formation and at the depth and temperatures of interest.

(c) The 58-32 well site is located within the 1.9 square mile Forge deep drilling site west of the Mineral Mountains, it is 217 miles south of Salt Lake City and 10 miles north-northeast of the town of Milford (Figure 5.2).

(d) A 20in conductor casing was to be set at 40 ft, a 17-1/2in surface hole with cemented 13-3/8in casing was planned to 300 ft, the intermediate 12-1/4in hole with cemented 9-5/8in casing was planned to 2,100 ft, below this an 8-3/4in hole to 7,060 ft was planned, with cemented 7 in casing to 6,900 ft, and open hole to total depth (TD).

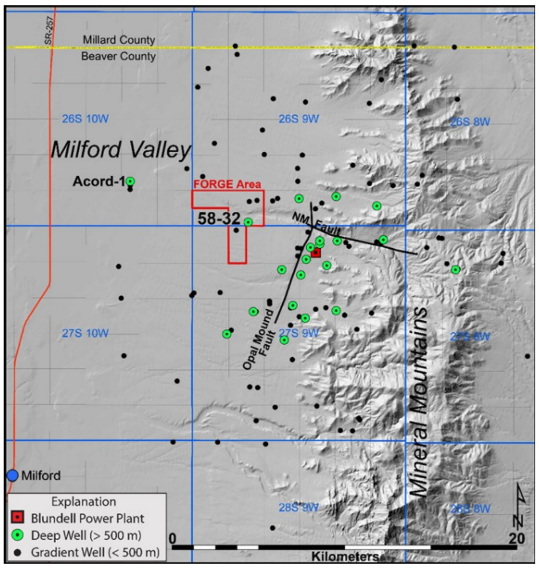


FIGURE 5.2: Location of Well 58-32 Milford, Utah (USA) [10] delimiting the area of interest.

The 58-32 well is characterized by three formations *Alluvium* (**F1**), *Granitoid "quartz-poor"* (**F2**) and *Granitoid "quartz-rich"* (**F3**). Detailed information is presented Figure 5.3.

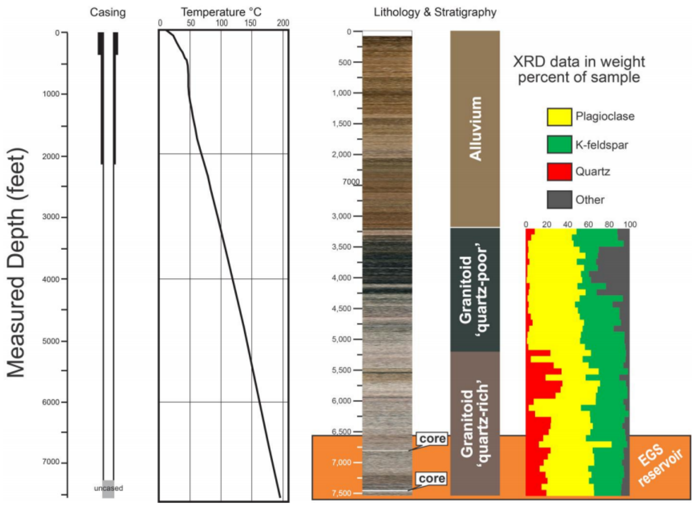


FIGURE 5.3: Lithology and stratigraphy of Well 58-32 [23]

Secondly, about **Well 21-31 Fallon, Nevada (USA)** [54] we have:

(a) Well 21-31 was drilled in 25 days (February 2018) to a depth of 6,108 ft.

(b) The 21-31 well site is located in Carson Field (Nevada) (Figure 5.4) with geographical coordinates: $39^o\, 23'\, 11.8114''N$ and $118^o\, 40'\, 00.1716''W$.

(c) A 20in conductor casing was to be set at 100 ft, a 17-1/2 in surface hole with cemented 13-3/8in casing was planned to 900 ft, below this an 12-1/4in hole was planned to 6,058 ft.

(d) The 58-32 well is characterized by four formations *neogene basin fill* (**F1**), *miocene basaltic andesite* (**F2**), *miocene rhyodacite* (**F3**), and *miocene basaltic andesite* (**F4**).

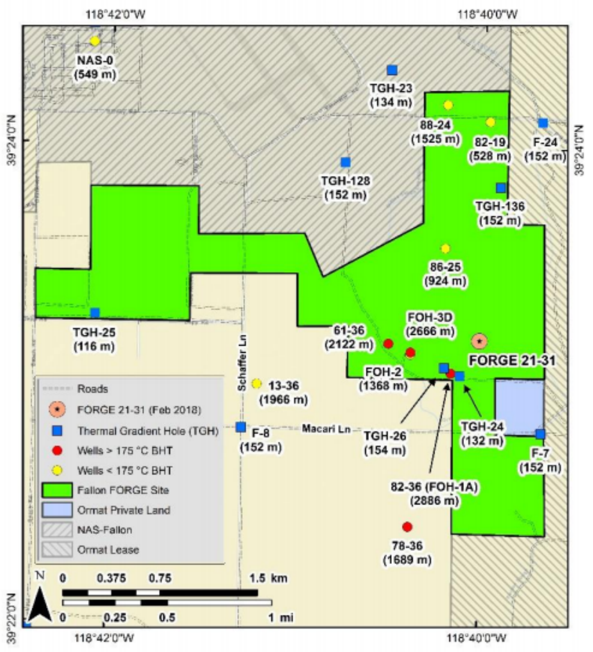


FIGURE 5.4: Location of Well 21-31 Fallon, Nevada (USA) [40]

The first analysis was developed with the data set of Well 58-32 [46]. This data set includes the following parameters: *ROP(ft/hr)*, *depth (ft)*, *weight on bit (k-lbs meaning kilo-pounds)*, *surface torque (psi)*, *rotary speed (rpm)*, *flow in (gal/min)*, *hookload (k-lbs)*, *temp out (^oF)*, *temp in (^oF)*, *pit total (bbls)*, *pump press (psi)*, *flow out %*, *wellhead pressure (psi)*, and partial pressure of H_2S with *H_2S floor (psi)*, *H_2S cellar (psi)*, *H_2S pits (psi)*. The information is visualized in Figure 5.5 which correspond to parameters normally monitored in a real-time operational center.

The analysis developed in this research will contain **static data evaluation** and experiments analysis **simulating real time** assessment using batches. In the case of *static data*, we assume that all information is already available. With this assumption, we want to examine the features available and the correlation among this features (Figure 5.6). *Dark shades* means positive correlation, *light shade* means negative. The stronger the color, the larger the positive correlation magnitude.

Correlation describes the association between variables. According to Figure 5.6, *rotary Speed (rpm)*, *flow in (gal/min)*, *pit total (bbls)*, partial pressure *H_2S Floor (psi)* has a positive correlation with *ROP (ft/hr)*. At the same time, *weight on bit (k-lbs)*, *temp out (^oF)*, *pump press (psi)*, *hookload (k-lbs)* and partial pressure *H_2S Cellar (psi)* has a strong correlation with *depth (ft)*. On the other hand, the physical model proposed by Bourgoyne and Young [58] defines that *ROP* depends on *diameter of the hole*, *weight on bit*, *rotary speed* and *flow*. This variables would be analyzed later.

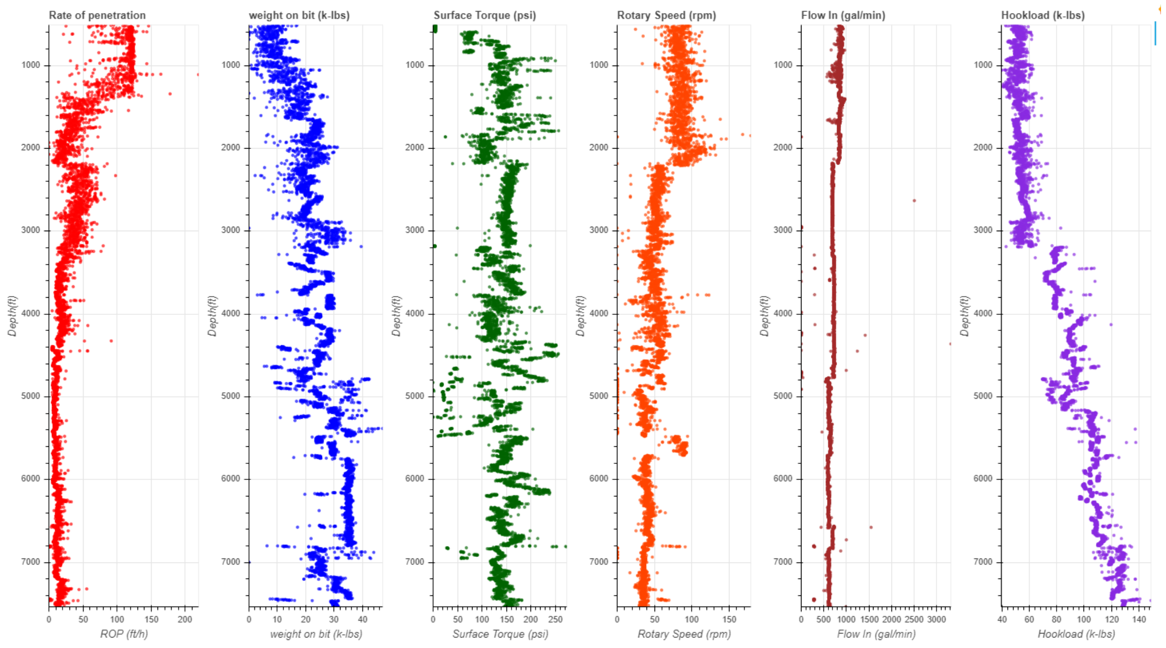


FIGURE 5.5: Parameters of drilling operation Well 58-32. *ROP (ft/hr)*, *weight on bit (k-lbs)*, *surface torque (psi)*, *rotary speed (rpm)*, *flow in (gal/min)*, and *hookload (k-lbs)*

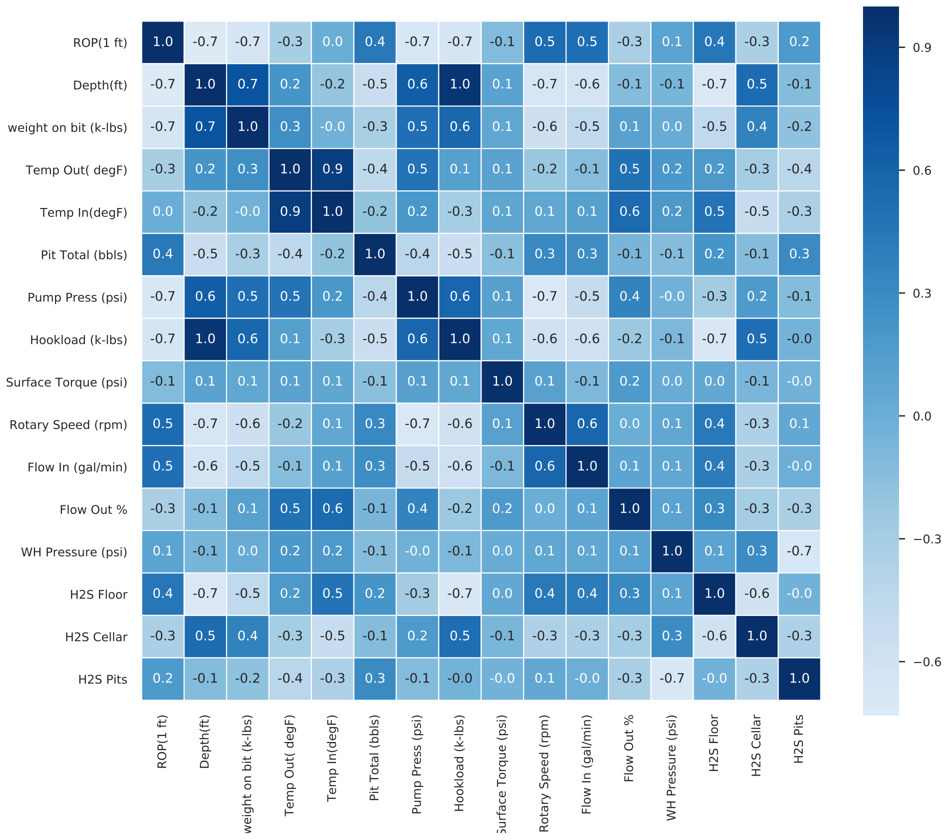
Moreover, *is our data set balanced?* It means, that the data set analyzed does not have any disproportion in the range of values that could affect future analysis. In Figure 5.7 we can inspect the distribution of our data set. The range of *hookload (k-lbs)* overpass the rest of the variables. These wide difference can impact in the machine learning model implementation. The use of support vector machine, or linear models on such a data (without normalization) will be dominated by *hookload (k-lbs)* and the rest of the variables will be almost ignored. This ensures that the different attributes are given an equal level of importance.

Therefore, we add the step of normalization presented in Figure 5.8. Normalizing data set, we subtract the mean in each sample $X := X - \mu$ where $\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$ and we divide samples by σ^2 where variance is defined as $\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)}\right)^2$.

With this assessment, we finished the characterization of our Well 58-32 data set considered in the context of *static* analysis. Now we are going to evaluate the implementation of machine learning models in the following sections.

5.2 Experimental results of machine learning implementation

In first place, we developed experiments based on *static data set* analysis. For this reason, the analysis of predictions is developed splitting all the data randomly between training (70%) and test (30%). The data is trained in the model selected (*support vector regression*, *random forest*, *xgboost* and *neural networks*). The data set for *support vector regression* and *neural networks* is standardized to zero mean and variance equal to one to remove effects of differing feature magnitudes, similar to the research developed by Soares et al [58].



	ROP(1 ft)	Depth(ft)	weight on bit (k-lbs)	Temp Out(degF)	Temp In(degF)	Pit Total (bbls)	Pump Press (psi)	Hookload (k-lbs)	Surface Torque (psi)	Rotary Speed (rpm)	Flow In (gal/min)	Flow Out %	WH Pressure (psi)	H2S Floor	H2S Cellar	H2S Pits
ROP(1 ft)	1.0	-0.7	-0.7	-0.3	0.0	0.4	-0.7	-0.7	-0.1	0.5	0.5	-0.3	0.1	0.4	-0.3	0.2
Depth(ft)	-0.7	1.0	0.7	0.2	-0.2	-0.5	0.6	1.0	0.1	-0.7	-0.6	-0.1	-0.1	-0.7	0.5	-0.1
weight on bit (k-lbs)	-0.7	0.7	1.0	0.3	-0.0	-0.3	0.5	0.6	0.1	-0.6	-0.5	0.1	0.0	-0.5	0.4	-0.2
Temp Out(degF)	-0.3	0.2	0.3	1.0	0.9	-0.4	0.5	0.1	0.1	-0.2	-0.1	0.5	0.2	0.2	-0.3	-0.4
Temp In(degF)	0.0	-0.2	-0.0	0.9	1.0	-0.2	0.2	-0.3	0.1	0.1	0.1	0.6	0.2	0.5	-0.5	-0.3
Pit Total (bbls)	0.4	-0.5	-0.3	-0.4	-0.2	1.0	-0.4	-0.5	-0.1	0.3	0.3	-0.1	-0.1	0.2	-0.1	0.3
Pump Press (psi)	-0.7	0.6	0.5	0.5	0.2	-0.4	1.0	0.6	0.1	-0.7	-0.5	0.4	-0.0	-0.3	0.2	-0.1
Hookload (k-lbs)	-0.7	1.0	0.6	0.1	-0.3	-0.5	0.6	1.0	0.1	-0.6	-0.6	-0.2	-0.1	-0.7	0.5	-0.0
Surface Torque (psi)	-0.1	0.1	0.1	0.1	0.1	-0.1	0.1	0.1	1.0	0.1	-0.1	0.2	0.0	0.0	-0.1	-0.0
Rotary Speed (rpm)	0.5	-0.7	-0.6	-0.2	0.1	0.3	-0.7	-0.6	0.1	1.0	0.6	0.0	0.1	0.4	-0.3	0.1
Flow In (gal/min)	0.5	-0.6	-0.5	-0.1	0.1	0.3	-0.5	-0.6	-0.1	0.6	1.0	0.1	0.1	0.4	-0.3	-0.0
Flow Out %	-0.3	-0.1	0.1	0.5	0.6	-0.1	0.4	-0.2	0.2	0.0	0.1	1.0	0.1	0.3	-0.3	-0.3
WH Pressure (psi)	0.1	-0.1	0.0	0.2	0.2	-0.1	-0.0	-0.1	0.0	0.1	0.1	0.1	1.0	0.1	0.3	-0.7
H2S Floor	0.4	-0.7	-0.5	0.2	0.5	0.2	-0.3	-0.7	0.0	0.4	0.4	0.3	0.1	1.0	-0.6	-0.0
H2S Cellar	-0.3	0.5	0.4	-0.3	-0.5	-0.1	0.2	0.5	-0.1	-0.3	-0.3	-0.3	0.3	-0.6	1.0	-0.3
H2S Pits	0.2	-0.1	-0.2	-0.4	-0.3	0.3	-0.1	-0.0	-0.0	0.1	-0.0	-0.3	-0.7	-0.0	-0.3	1.0

FIGURE 5.6: Correlation of drilling operation parameters Well 58-32 (*16 features*) where *Dark shades* means positive correlation, *light shades* means negative correlation.

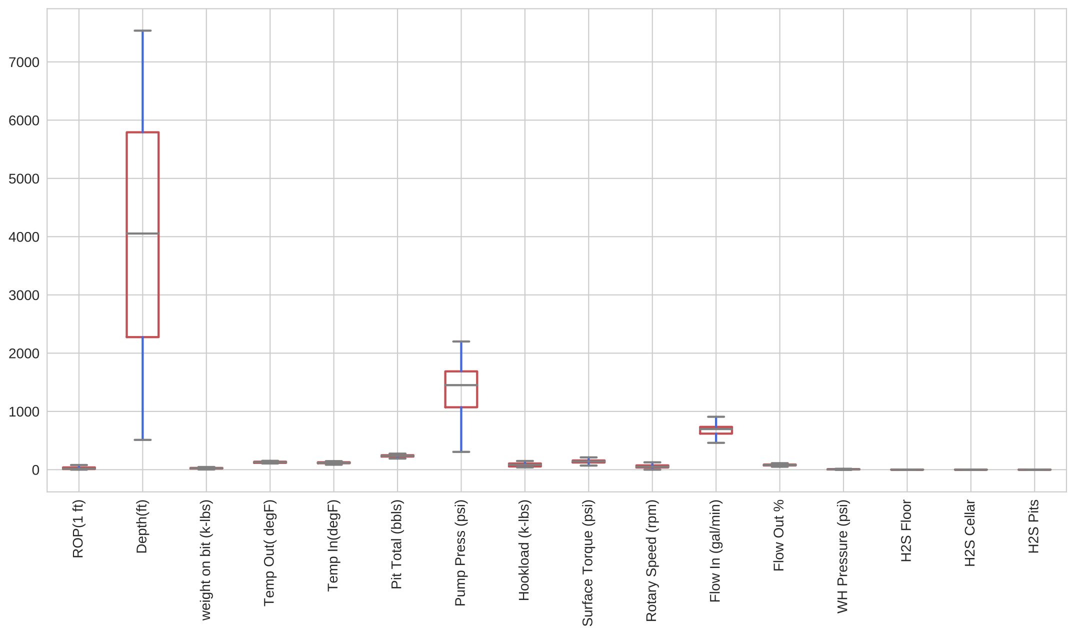


FIGURE 5.7: Boxplot representing the distribution of *16 features* in Well 58-32

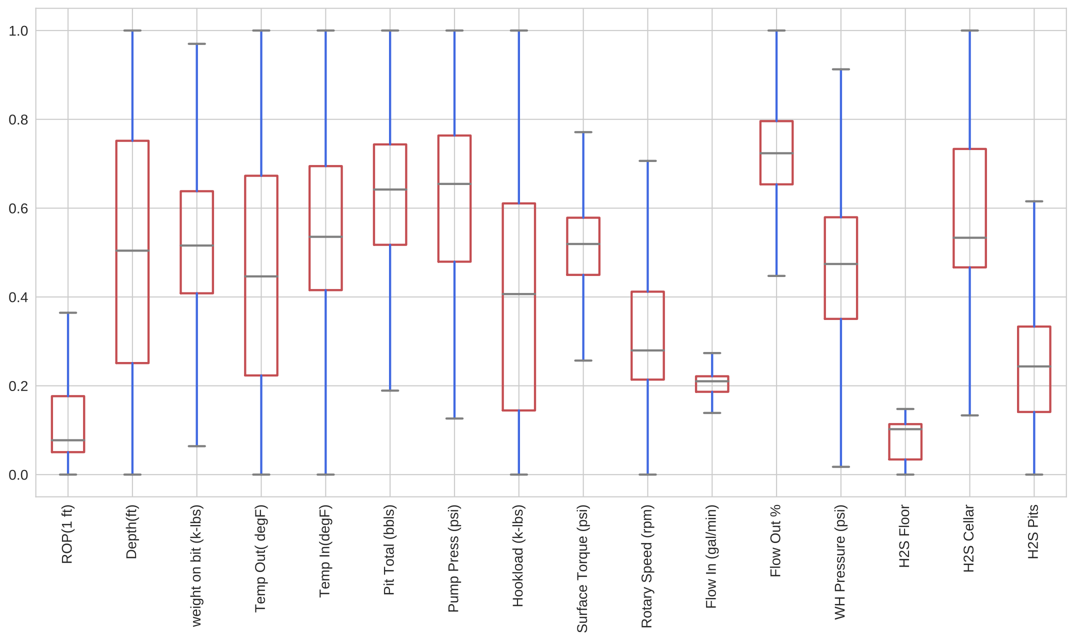


FIGURE 5.8: Boxplot representing the normalized distribution of *16 features* in Well 58-32

In order to optimize the models, *what is the hyper-parameter combination that achieve better performance in each technique?* Following the structure of Figure 4.3 we are able to identify the parameters. The experiment is implemented with *cross-validation* analysis which involves randomly $k - fold$ equal size to identify the best parameters per model. Previously assessment, we defined the grid of option for each hyper-parameter. The grid of hyper-parameters and the respective best options are presented in Table 5.1, Table 5.2, Table 5.3, and Table 5.4. The models were run with 20 iterations and cross validation *k-fold* equal to 4.

TABLE 5.1: Hyperparameters alternatives for *random forest*

Hyperparameter	**Grid**	**Best hyperparameter**
Number of trees	(2 to 2000 step 20)	632
Maximum features per split	(2, 5, 10, 15)	5
Minimum sample to split	(2, 3, 4, 5, 10)	3
Minimum sample per leaf	(1, 2, 4)	2
Maximim depth	(4 to 30 steps 2)	30

TABLE 5.2: Hyperparameters alternatives for *support vector regression*

Hyperparameter	**Grid**	**Best hyperparameter**
Kernel function	Linear, Polynomial (3rd degree), RBF, Sigmoid	RBF
Epsilon (ε)	(0.01, 0.1, 1, 10)	0.1
Penalty parameter C	(1, 10, 100, 1000)	1
Kernel coefficient (γ)	(0.0001, 0.001, 0.01, 0.1)	0.1

TABLE 5.3: Hyperparameters alternatives for *XGBoost*

Hyperparameter	**Grid**	**Best hyperparameter**
Number of estimators	(2 to 2000 step 20)	1474
Maximum depth	(3,4,5,6,7,8,9,10)	9
Learning rate	(0.0001, 0.001, 0.01, 0.1)	0.01
Minimum child weight	(2,4,6,8,10,12)	12
Gamma	(0.0, 0.1, 0.2, 0.3, 0.4)	0.4
Subsample	(0.7,0.8,0.9)	0.8

As aforementioned, we obtained the hyper-parameters that will contribute to achieve a good prediction. Therefore, *What is the model with the best performance?*. The performance indicators used to evaluate the models correspond to *Root mean squared error*. Table 5.5 summarize the results for each machine learning algorithm.

The machine learning algorithm with the lowest $RMSE$ was *random forest* with a close result obtained by *xgboost*. This result matched with the studies developed by Hegde et al [30] [31]. The low deviation obtained by *Random forest* proved that the ensemble method are competent for rate of penetration prediction. In addition, we wanted to evaluated how was the improvement of the models after selecting the

TABLE 5.4: Hyperparameters alternatives for *neural network*

Hyperparameter	**Grid**	**Best hyperparameter**
Optimizer	Adam, SGD, LBFGS	LBFGS
Number of neurons in hidden layer	(2, 3, 4, (2,1), (2,2), (3,1), (3,2), (3,3))	(3, 1)
Activation function	Identity, logistic, Tanh, ReLu	ReLu
Regularization	(0.00001, 0.0001, 0.001, 0.01, 0.1)	0.0001

TABLE 5.5: Performance of machine learning models in *stationary analysis* according to RMSE and normalized RMSE

Method	**RMSE before optimization**	**RMSE after optimization**	**Normalized RMSE**
Random forest	9,99	9,57	0,29
XGBoost	10,36	9,8	0,30
Support vector regression	45,88	10,58	0,32
Neural networks regression	10,87	11,46	0,35

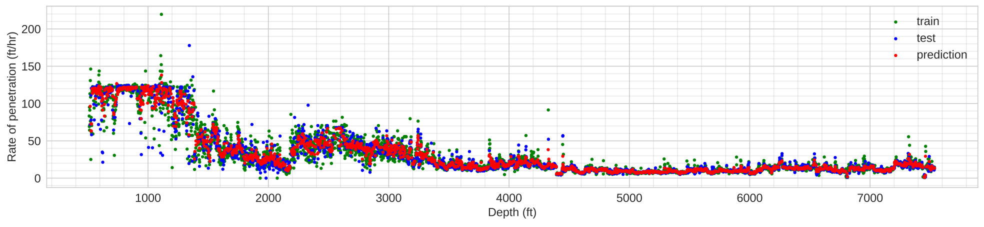
best hyperparameters. Therefore, *random forest* represented an improvement of 4.2% and *xgboost* achieved 5.4%. Meanwhile, *support vector regression* improved 77% and *neural networks* decreased 5.4%, probably overfitting. *Support vector regression* will require high computational and manual demand tuning the model. The visualization of prediction is presented in Figure 5.9.

Random forest offers the possibility to analyse the ranking of the main features that contribute to predict our objective value (rate of penetration). Therefore, *What are the features that describe the 95% of importance in our data set?* The ranking of features is presented in Figure 5.10. Using the *gini importance* [15] model with *mean decrease in impurity (MDI)* explained in section 4, we obtained the importance of each feature (Figure 5.10 and Figure 5.11).

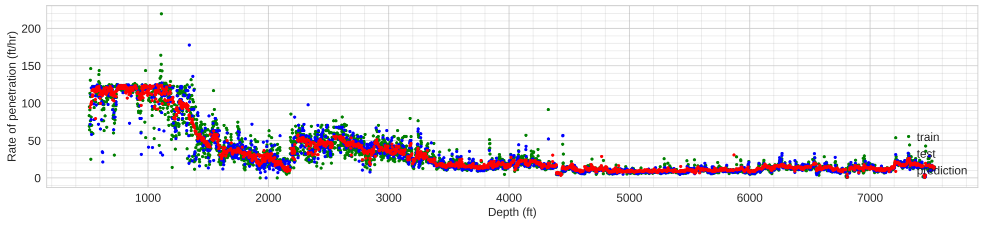
The main feature selected are: *depth*, *hookload*, *surface torque*, *flow in*, *flow out*. Some of the features as *rotary speed* and *weight on bit* were not identify by the model as an important variable for prediction of *rate of penetration*. This results contrast with the reality of operation where *rotary speed* and *weight on bit* are controlled parameters in the drilling rig.

However, with this unexpected result, *What is the impact in the predictions if we reduce the number of features?* To have a complete analysis, we can assess *random forest* with the total number of features available and compare the performance once we reduce the number of variables. The results were obtained after 10 iterations. Based on *gini importance*, we collected the main features, in this case 5 features, to train and test *random forest*. The results are presented in Table 5.6. The analysis did not include parameters optimization.

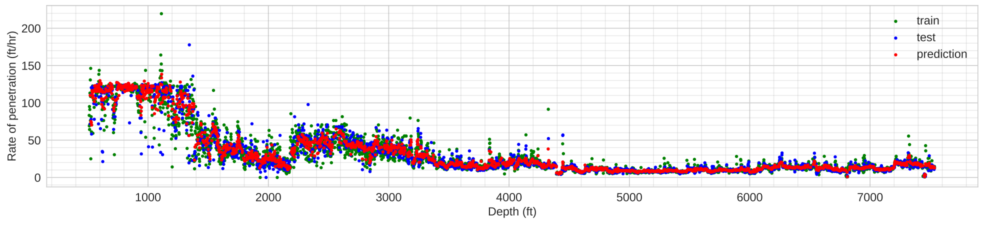
Reducing the number of features does not represent a relevant impact in the performance of *random forest*, indicating that we could work with all the features. In fact, the *root mean squared error* increased. The only benefit that we obtained



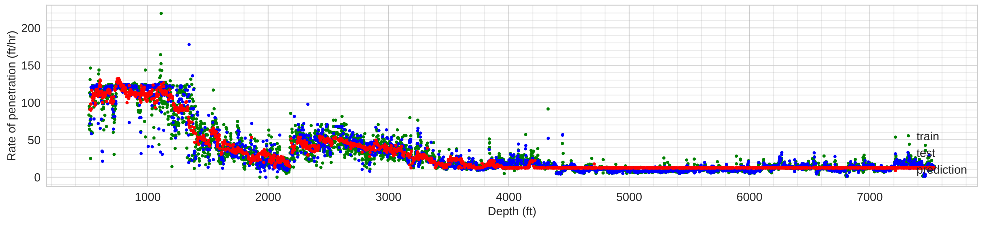
(A) Prediction with *random forest*



(B) Prediction with *support vector regression*



(C) Prediction with *extreme gradient boosting*



(D) Prediction with *neural networks*

FIGURE 5.9: Visualization of *rate of penetration* predictions using *support vector regression*, *random forest*, *xgboost* and *neural networks* in Well 58-32

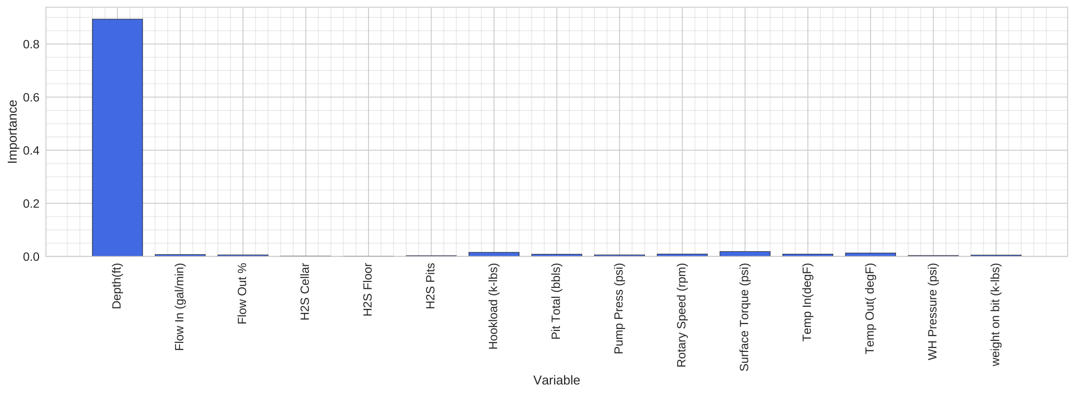


FIGURE 5.10: Feature importance using *random forest*

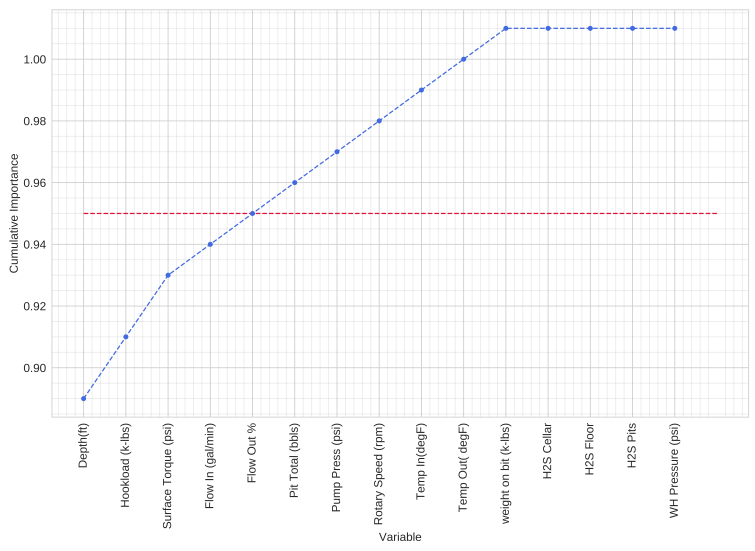


FIGURE 5.11: Cumulative importance using *random forest*

TABLE 5.6: Performance of *random forest* reducing the number of features in Well 58-32

Number of features	**RMSE**	**Run time (sec)**
16 features	9,99	0,46
5 features	10,23	0,18

is that the run time is reduced in 0.28*sec*. In addition, the analysis was developed with only the features considered for control parameters by Soares et al [58] (weight on bit, drilling rotational speed, and drilling fluid (mud) flow rate) and depth. On this run, we obtained a performance (RMSE) of 10.28 ft/hr which is higher that the performance with all features (9,99 ft/hr). Still, we need to work with a realistic model to assess the real-time operation. This analysis will come later. In this way, we finish the evaluation in the context of *static data* assuming that we have available all the data after operation. This scenario only contribute to analyze what models are competent to offer a good prediction, identify the main features and obtain optimized hyper-parameters.

Now, we need to approximate our assessment to a more realistic scenario. Therefore, *what is the performance of machine learning models splitting the data set per batches increasing the data available?* With this scenario, we are trying to assess the option increasing the data available to be trained for our model. This experiment was developed under the following conditions:

(a) We analyzed *linear regression*, *k-neighbors regression*, *support vector regression*, *decision trees*, *random forest*, *xgboost*, *ada boost regression* and *neural network regression*.

(b) Nine (9) batches were implemented and we tested our model in 266 points (Table 5.7).

(c) The performance was measured based on *root mean squared error* (Figure 5.12). For each batch, this performance indicator is appended.

TABLE 5.7: Definition of number of points for each batch split in train-test set for Well 58-32

Batch	Train set	Test set
1	691	691
2	1382	691
3	2073	691
4	2764	691
5	3455	691
6	4146	691
7	4837	691
8	5528	691
9	6219	691

After training and testing each model, the performance and run time are represented in the box plot Figure 5.12 and Table A.1, Table A.2 and Table A.3. *Random forest* endorsed its capacity achieving lower average error (RMSE:14.55) evaluated in each batch. *Decision tree* achieved the second lowest average error (RMSE:16.75) whit high deviation. *Xgboost* and *k-nearest neighbor* obtained a compact distribution similar to *random forest*.

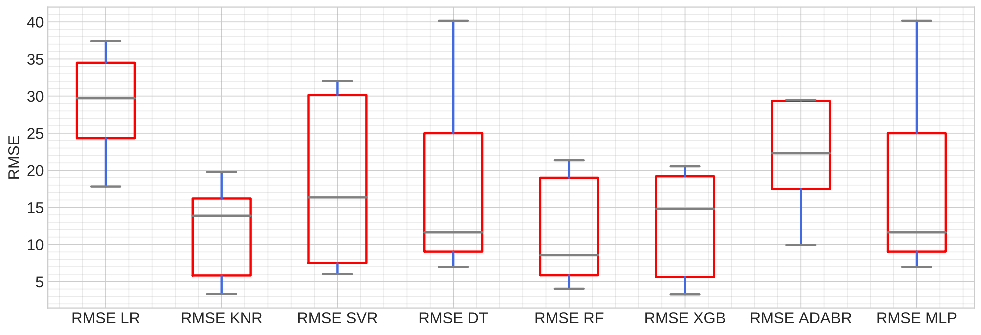
What is the model that run the respective algorithm with the lowest time?. According to the results presented in Figure 5.12 the model with the lowest run time average was *linear regression* with 0.002 sec and *k-nearest neighbor* with 0.043 sec (Figure 5.12). On the other hand, *neural networks* had the highest run time average among the models (2.98 sec). Still, we need to take into account the option of *neural networks* which has a lot of potential in future research related to *reinforcement learning*. According to Aggarwal [3], *neural networks* are slow to train. Computational complexity is an inherent problem with neural networks. Nevertheless, recent hardware and algorithmic advancements have made neural networks (in general) and deep learning (in particular) more feasible [3].

Being *random forest* regression the model with the better performance among the models, generating good generalization performance in most cases at speeds much faster than the rest of the learning algorithms. This model obtained a normalized average RMSE equal to 0.59 which is higher compared to stationary analysis (Normalized RMSE equal to 0.29 with *random forest*). We were interested to observe the behavior of predictions per batch. Therefore, in Figure 5.13 is presented the results.

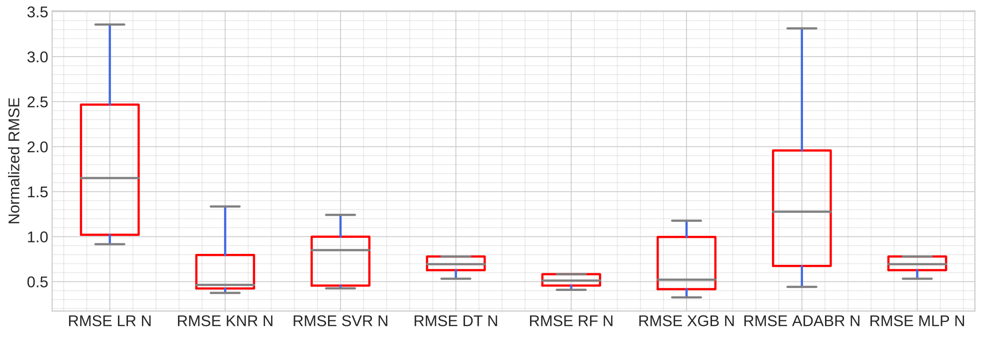
An additional approach that we can explore based on the proposal of Soares [58] and Helge [30] is to implement the machine learning model per formation. According to these authors, the lithology-dependence assumption for physical ROP models is maintained for machine learning techniques. In the following analysis, we are going to use the data set available from Well 58-32. Based on the tops of each formation presented in Figure 5.3. The characterization of the data set is presented in Table 5.8.

To contribute in our characterization, we can observe the distribution of *rate of penetration* in the different formations (Figure 5.14).*F1* has the highest standard deviation with a pick in 125 ft/hr. Meanwhile, *F1* and *F2* were described with similar behavior. The number of bins were defined automatically by the system.

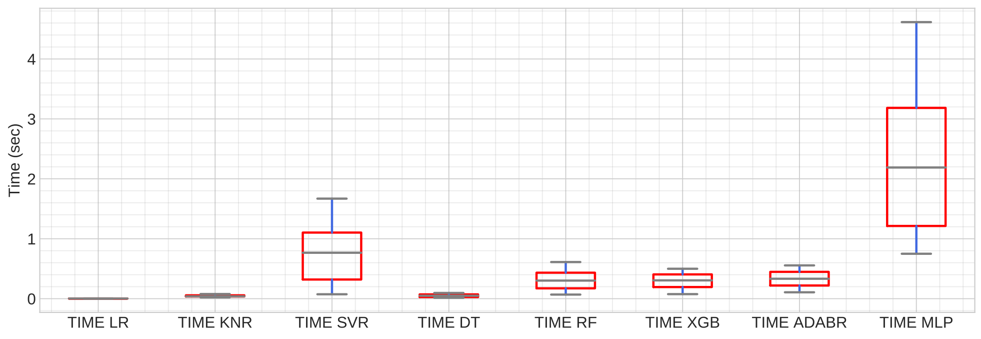
Simulating the incremental availability of data, the model is trained with the initial points and the test were executed with the remaining points according to the distribution presented in Table 5.9 for Well 58-32.



(A) RMSE for each machine learning algorithm



(B) Normalized RMSE for each machine learning algorithm



(C) Run time (sec) for each machine learning algorithm

FIGURE 5.12: Performance (RMSE and time) of each machine learning model increasing the data available (Batch) in all data set.

What was the performance of the machine learning techniques using incremental data available?. To answer this question, we obtained the normalized metrics for each iteration which could be presented in the box plot Figure 5.15, Table A.4, Table A.5, and Table A.6.

Among these algorithms, *random forest* and *xgboost* endorsed its capacity to obtain accurate predictions in Formation 1. As well as *k-nearest neighbor*. However, *k-nearest neighbor regression* developed better average performance in Formation 2 and 3. These performance is better compared to batch analysis without formation where the average error with the best model (*random forest*) was 0.59 with the exception of the results in Formation 2. In Figure 5.16, we visualized the prediction in Formation 1 for Well 58-32 increasing the training data.

The same analysis was developed for 21-32 Well data set. The characterization of each formation is presented in Table 5.10.

In addition, we included the distribution of *rate of penetration* per formation in Figure 5.17. High standard deviation is observed in each of the Well 21-31 formations.

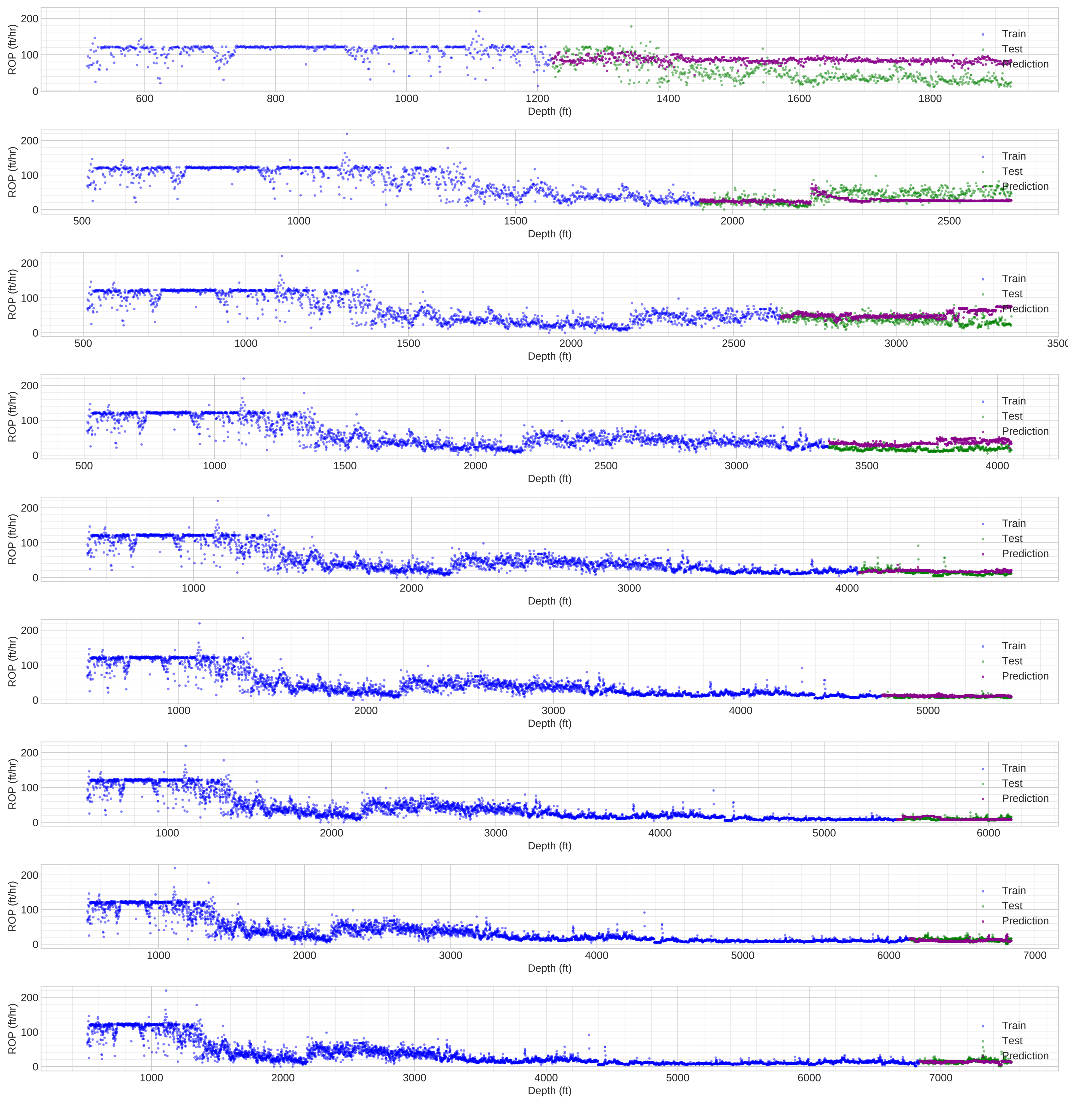
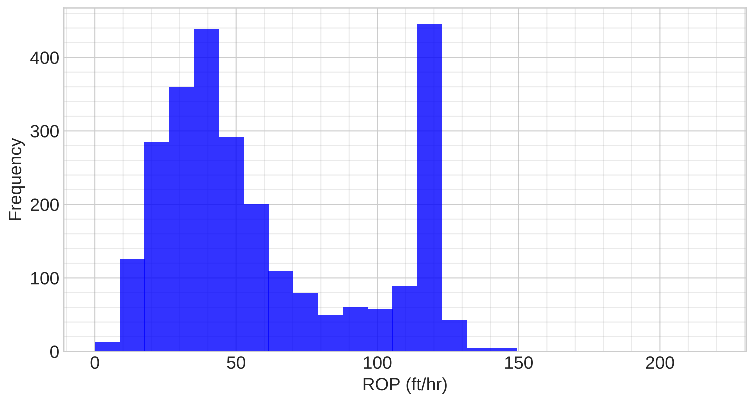


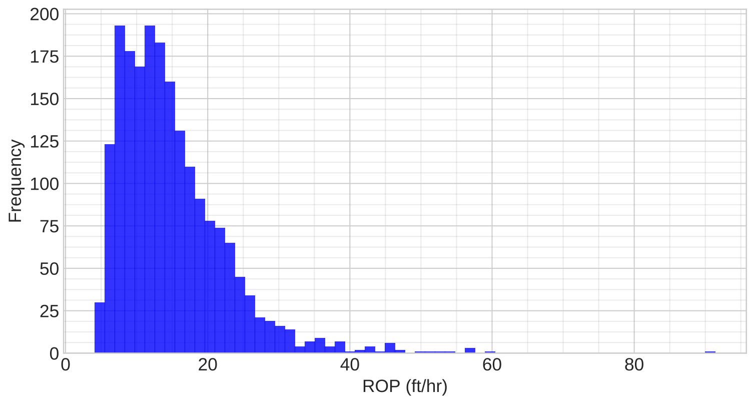
FIGURE 5.13: Visualization of training, test set and prediction Well 58-32 (9 Batches) with *random forest*

TABLE 5.8: Characterization of data set Utah Well 58-32 per formation

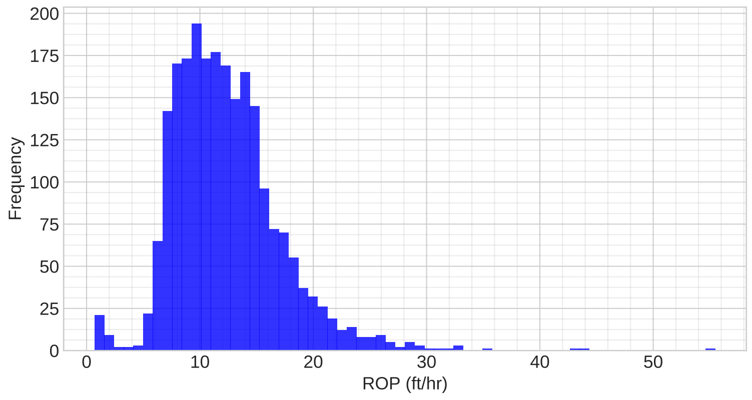
Formation	Thickness (ft)	Number of points	ROP average (ft/hr)
F1	3165	3064	60.59 ± 36.78
F2	2000	1983	14.92 ± 7.62
F3	2286	2264	12.33 ± 7.73



(A) Formation (**F1**)

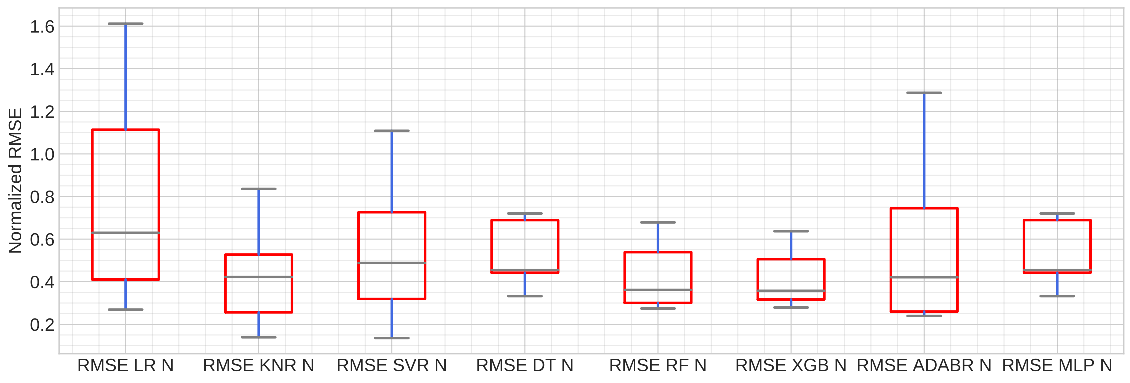


(B) Formation 2 (**F2**)

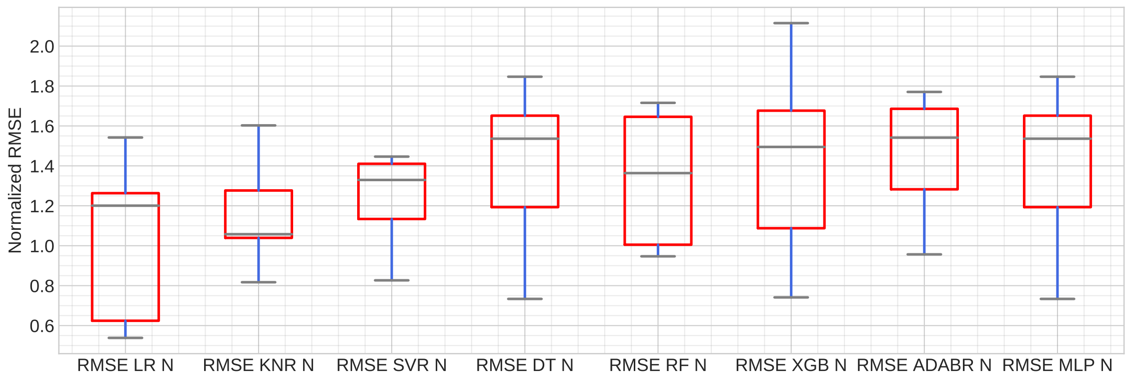


(C) Formation 3 (**F3**)

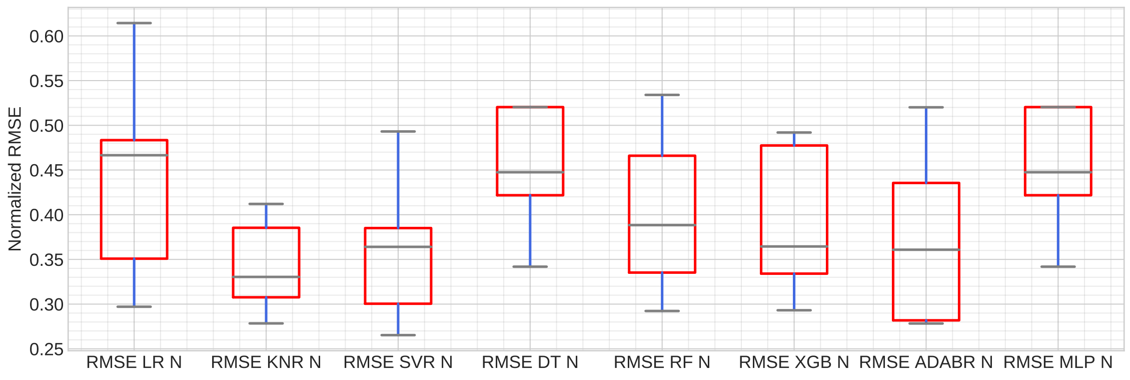
FIGURE 5.14: Histogram of *rate of penetration (ft/hr)* per formation
Well 58-32



(A) Formation (**F1**)



(B) Formation (**F2**)



(C) Formation (**F3**)

FIGURE 5.15: Performance of *rate of penetration (ft/hr)* prediction per formation with normalized RMSE in Well 58-32

TABLE 5.9: Distribution of points in train-test set per formation Well 58-32 distributed in 9 batches

Batch	Formation 1		Formation 2		Formation 3	
	Train set	Test set	Train set	Test set	Train set	Test set
1	269	266	201	198	230	226
2	535	266	399	198	456	226
3	801	266	597	198	682	226
4	1067	266	795	198	908	226
5	1333	266	993	198	1134	226
6	1599	266	1191	198	1360	226
7	1865	266	1389	198	1586	226
8	2131	266	1587	198	1812	226
9	2397	266	1785	198	2038	226

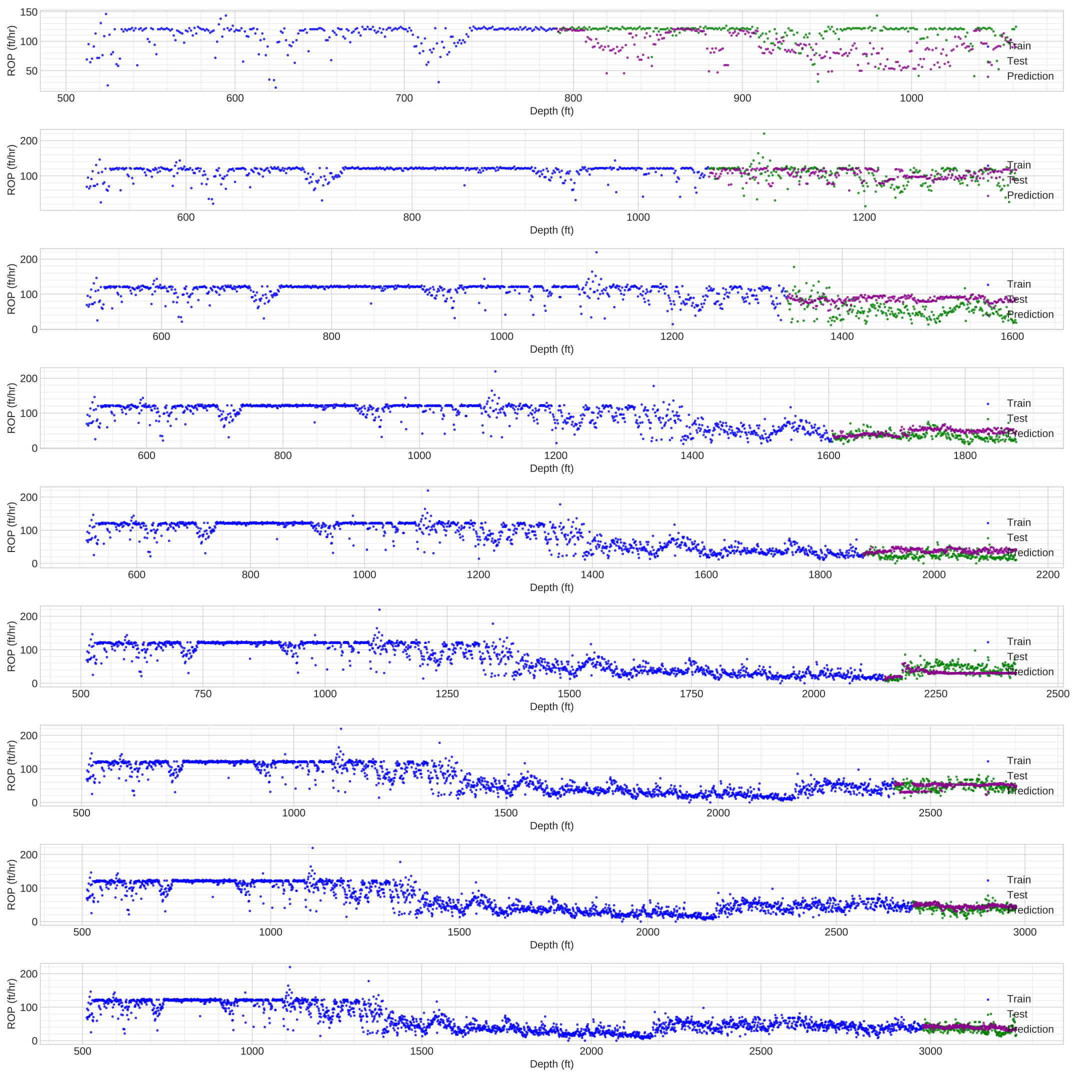
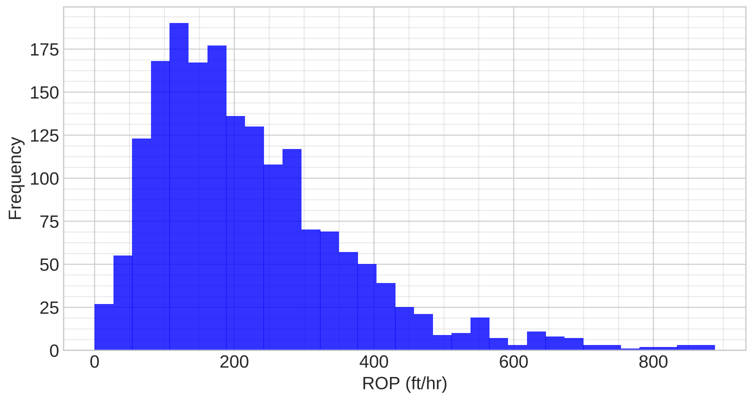
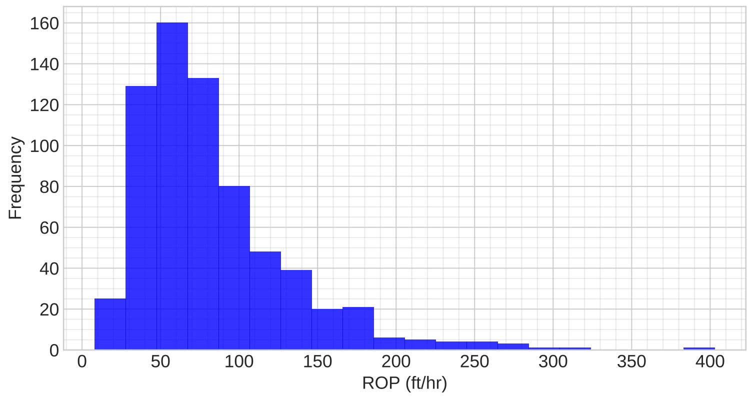


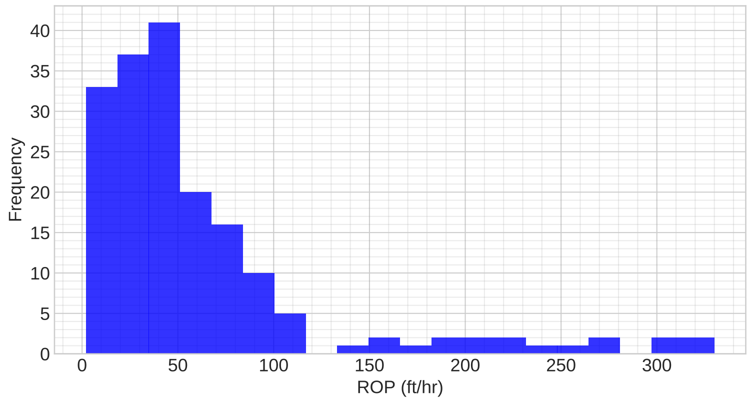
FIGURE 5.16: Visualization of train, test and prediction for Formation F2 in Well 58-32 (9 batches)



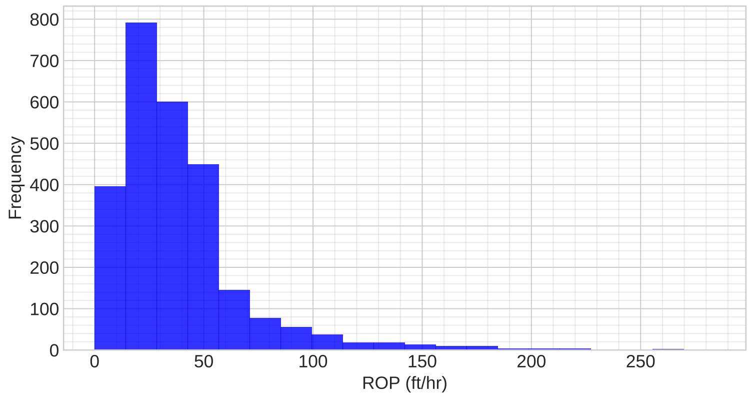
(A) Formation (**F1**)



(B) Formation (**F2**)



(C) Formation (**F3**)



(D) Formation (**F4**)

FIGURE 5.17: Histogram of *rate of penetration (ft/hr)* per formation Well 21-31

TABLE 5.10: Characterization of data set Fallon Well 21-31 per formation

Formation	Thickness (ft)	Number of points	ROP average (ft/hr)
F1	1950	1820	220.69 ± 140.45
F2	680	680	82.27 ± 48.96
F3	180	180	61.21 ± 65.34
F4	2635	2635	38.19 ± 30.81

After training and testing, we achieved the results presented in Figure 5.18 for each formation assessing *linear regression*, *k-nearest neighbor regression*, *support vector regression*, *decision tree*, *random forest*, *xgboost*, *adaboost* and *neural networks*.

We can observe that *support vector regression* achieved better performance in *Formation 1*. In *Formation 2*, *Linear regression* obtained lower error. Furthermore, *k-nearest neighbor regression* offered better score in *Formation 3* and *Formation 4*. However, it is important to noticed that the performance of prediction started decreased after batch 5 where $RMSE$ started to increase.

5.3 Experimental results of anomaly detection

In this section, we are going to explore the models selected to detect anomalies under the following conditions:

(a) The first scenario corresponds to **stationary evaluation** having all the data available. We assessed *k-means clustering*, *one-class support vector machine*, *isolation forest* and *autoencoder*. A total of *16 features* were analyzed including *rate of penetration*. In addition, with the exception of *autoencoder*, the machine learning models were evaluated in all the data set. In the case of *autoencoder*, we had to split the data in train and test set.

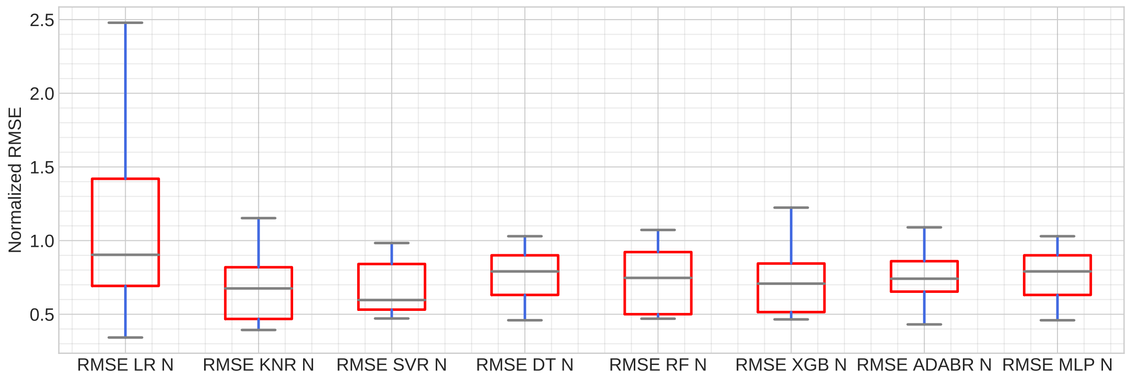
(b) The second scenario is developed with **regression models and distance of prediction**. This section includes: *random forest*, *xgboost*, *k-nearest neighbor*, *decision tree*, *support vector regression*, and *neural network*, the same models used in Section 1.

(c) Based on the recommendations by Aggarwal [3], an exploratory and visual analysis has been implemented in all stages of anomaly detection.

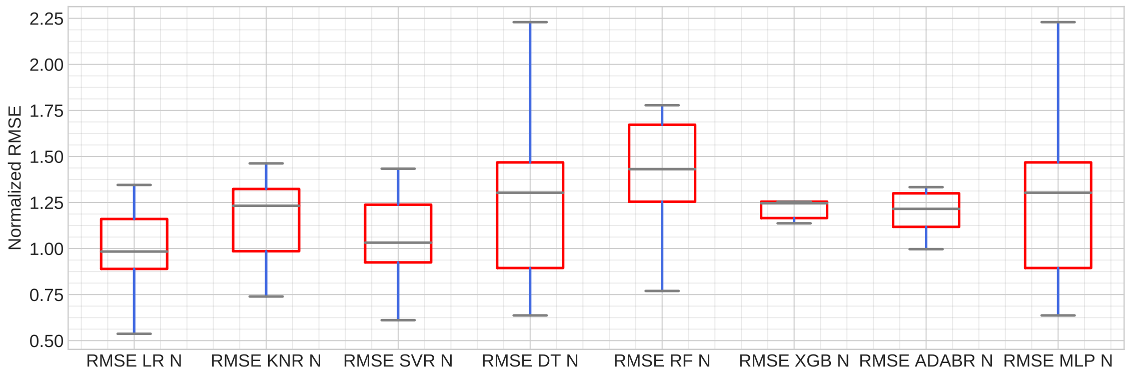
(d) The data set available (Well 58-32 and Well 21-31) does not include any label defining which points are considered anomaly or normal point. We are evaluating in unsupervised learning where we do not know which data point is normal and which one is an anomaly. The data set used for anomaly detection corresponds to Well 58-32.

(e) In general, the outlier fraction is defined in 0.01 (1%) for all the models. In further research, this value requires to be adjusted. With more data, we can define a more realistic threshold.

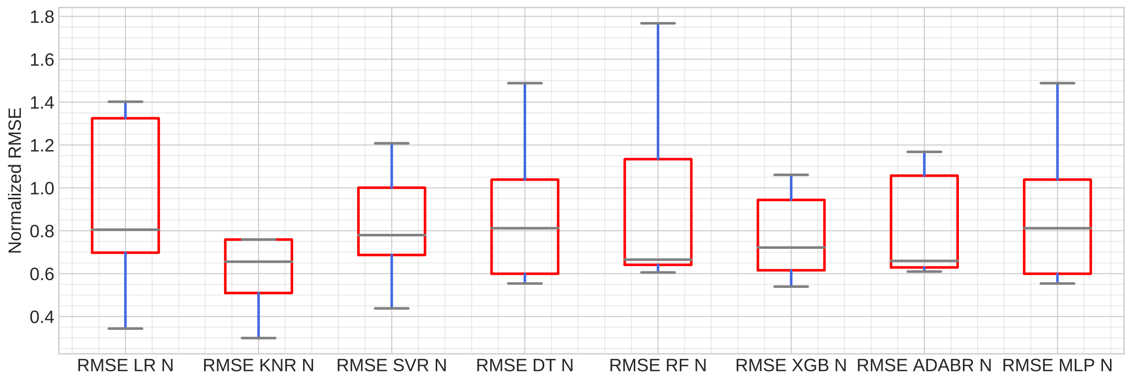
(f) Hyperparameter optimization is not included in our analysis. Part of this decision is based on the lack of anomaly labels. Moreover, setting hyper-parameters in anomaly detection is generally a difficult unsolved problem unless there is a



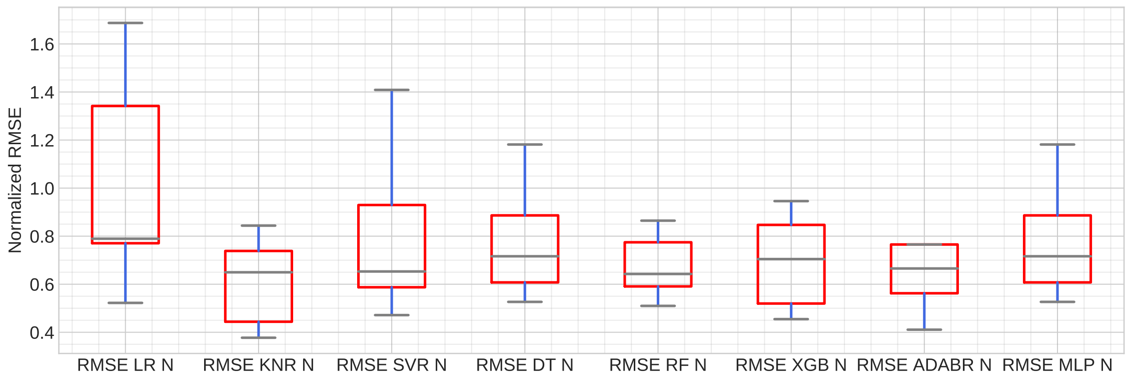
(A) Formation (**F1**)



(B) Formation (**F2**)



(C) Formation (**F3**)



(D) Formation (**F4**)

FIGURE 5.18: Performance of *rate of penetration (ft/hr)* prediction per formation with normalized RMSE in Well 21-31

validation ground truth available for the parameter tuning according to Pevny et al [49].

Therefore, the first method analysed in stationary scenario is **k-means clustering**. With clustering in *unsupervised learning*, non knowledge of anomaly is needed while we are training for detection [19]. On the other hand, classification methods needs to be trained with normal an abnormal data to be able to separate those classes during detection.

- As aforementioned in Chapter 4, we implemented *principal component analysis* reducing the dimensionality of our data (*16 features*) in two principal components. Our implementation of *principal component analysis* based on principal component transformation used two (2) components capturing more than 95% variance.
- Then, *elbow method* (Figure 5.19) has been used to determine the optimal number of clusters. We obtained the score of variance for each number of centroids assessed.

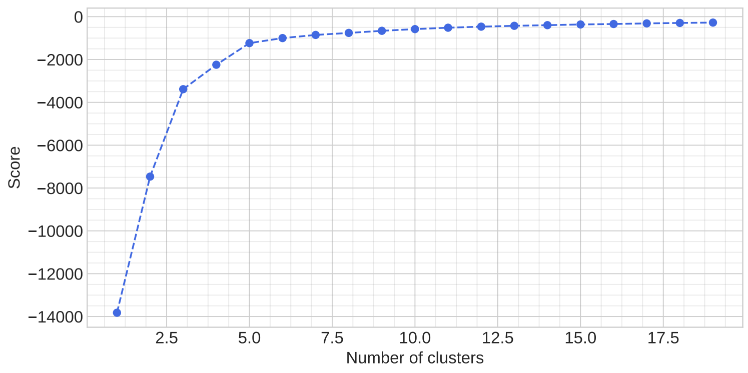


FIGURE 5.19: Elbow curve to determine number of clusters required for *k-means clustering* Well 58-32

We can observe from the *elbow curve* (Figure 5.19), the graph leveled off after 14 clusters. Large clusters correspond to normal data, and small clusters possibly correspond to anomalies. After we computed the distance between each point and its nearest centroid, the biggest distances were considered as anomaly. The result is presented in Figure 5.20 where *dark points* correspond to normal data and *light shades* represent anomalies.

However, we need to observe this information in our response parameter to have a reference of the anomaly location. It means, the algorithm has identified in one of the variables (*16 features*) a deviation. Figure 5.21 display the anomaly detection using *k-means clustering* technique. Finally, this method detected 69 anomalies in 6511 points.

The second method analysed in stationary scenario corresponds to **one-class support vector machine** used on *unsupervised detection*. The use of this method without normalization would be dominated by the feature *hookload* and the rest of the variables would be almost ignored. Normalization ensures that the different attributes are given an equal level of importance in the anomaly detection process. We trained all data set to infers the properties of normal cases and from these properties can predict which examples are unlike the normal examples. Then, the data is labeled

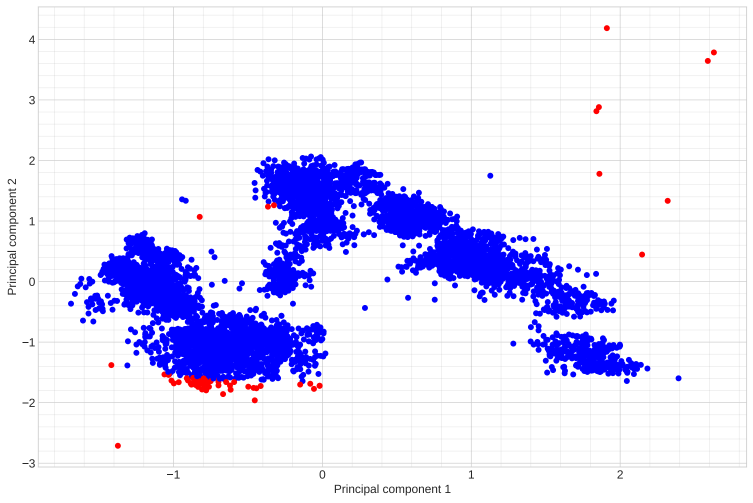


FIGURE 5.20: Visualization of two principal component identifying the anomalies Well 58-32

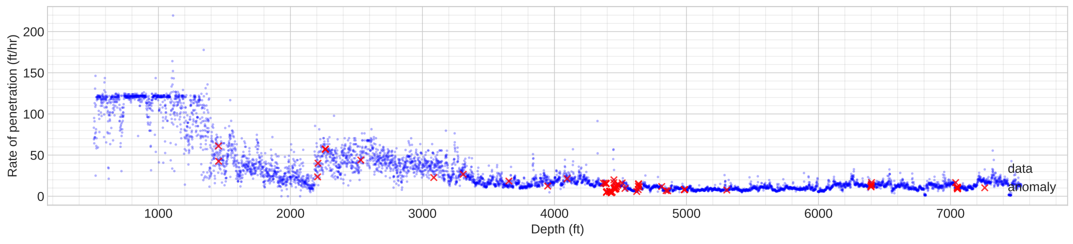


FIGURE 5.21: Anomaly detection using *k-means clustering* technique Well 58-32

as normal and anomaly which is represented in Figure 5.22. *One-class support vector machine* detected 69 anomalies and 6511 points.

The third method evaluated is **isolation forest**. As was explained in Chapter 3, this method select the shorter path to define the anomalies. Outlier points are usually isolated quickly with a few splits. The location of the anomalies are displayed in Figure 5.23. The method identified 69 anomalies in 6511 points.

In Table 5.11, you can find the total number of anomalies detected by each machine learning model. However, we can not define which model performed better. Further research will require the label of anomalies to evaluate the performance of each algorithm.

TABLE 5.11: Number of anomalies detected for each method Well 58-32

Method	**Run time (sec)**	**Normal**	**Anomaly**
Isolation forest	1,19	6459	52
One class support vector	0,12	6442	69
k-means clustering	15,88	6442	69

According to the results, *isolation forest* and *one-class support vector machine* methods performed favourably in terms of run-time. The cause of anomaly is not

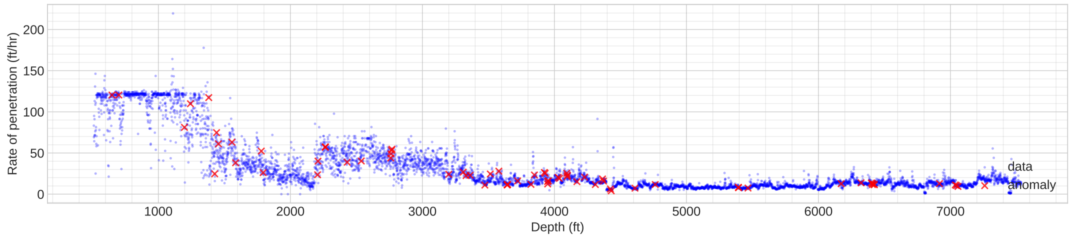


FIGURE 5.22: Anomaly detection using *one-class support vector* technique Well 58-32

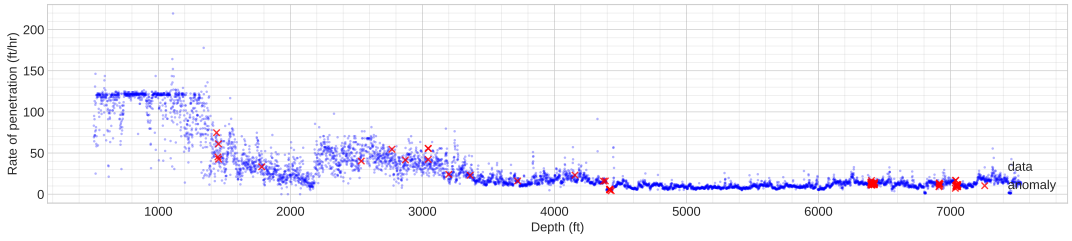


FIGURE 5.23: Anomaly detection using *random forest* technique Well 58-32

identified representing a drawback of the implementation. Explaining the causes of an anomaly detection is relatively new field and there is not yet an established methodology to evaluate and compare the quality of algorithms [49].

We have finished with the stationary analysis. Then, we explored the second scenario in anomaly detection, where we were predicting part of the interval. Then, we measure the difference between the prediction and measured data. In the assessment, we have defined a interval of analysis between 3595 ft to 4097 ft in Well 58-32 where 70% represents training and 30% testing. The proportion of training and test were defined in a sequence. To detect the anomalies, we need to define a threshold which represent the proportion of outliers (1.5%). However, this reference requires the analysis of several data set to define a common threshold. For this study, the limit has been arbitrarily defined. Further research, with more data available, will contribute to define a realistic limit. In Figure 5.24 training and test set are visualized.

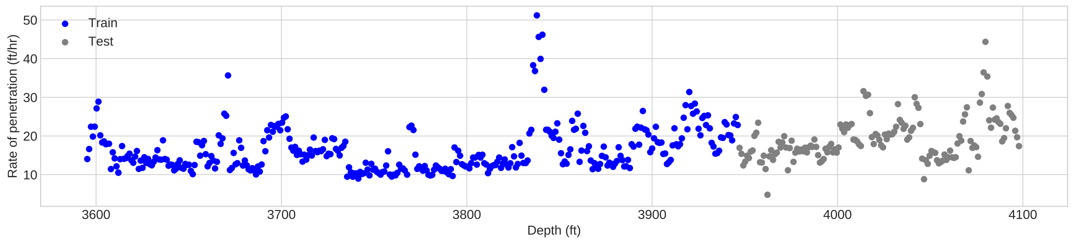
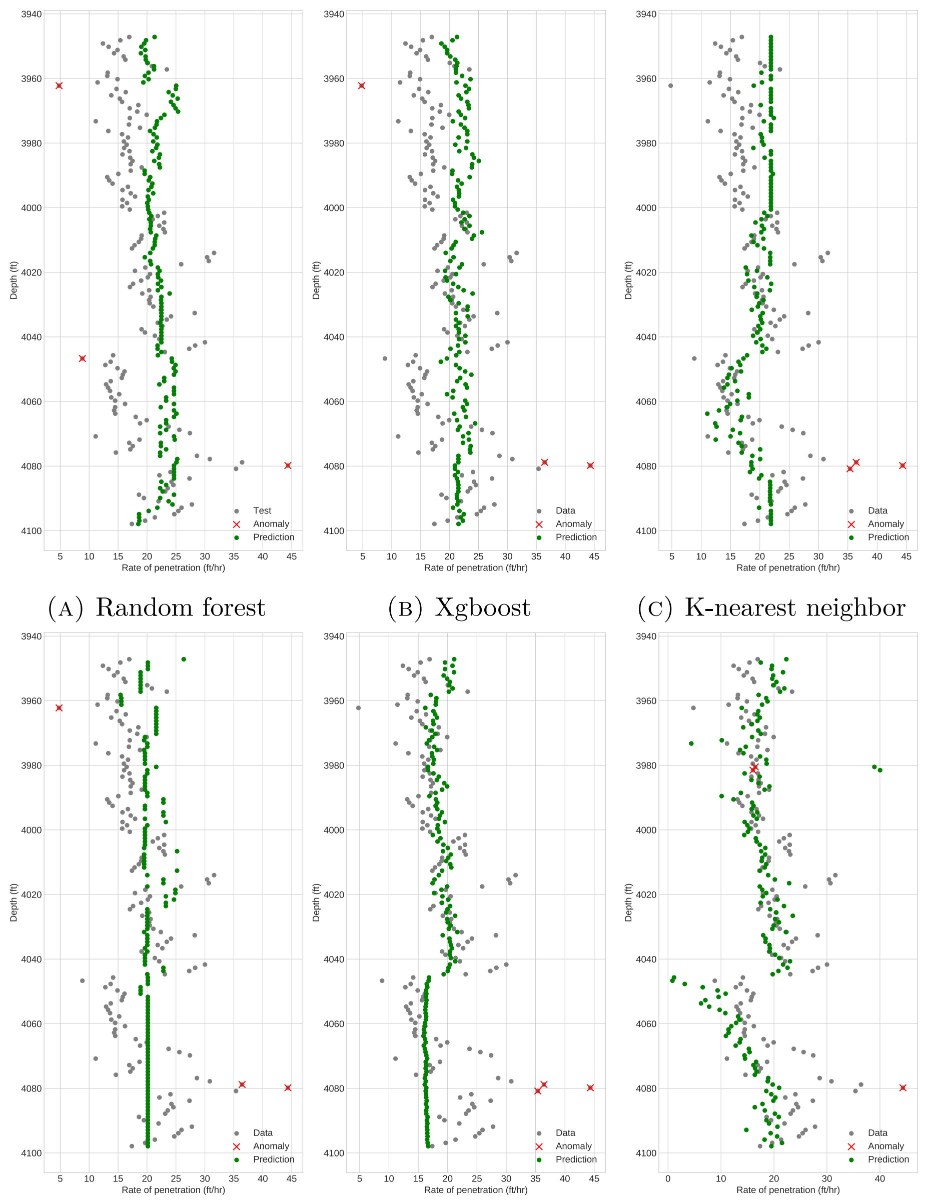


FIGURE 5.24: Train and test for prediction and anomaly detection between 3595 ft to 4097 ft Well 58-32

To predict the subsequent interval, different machine learning algorithm (*random forest*, *xgboost*, *k-nearest neighbor*, *decision tree*, *support vector regression*, and *neural network*) were implemented. In Figure 5.25 is presented the result for each algorithm.

shade points represent test data, *light points* correspond to predictions and *x points* perform as anomalies.



(A) Random forest (B) Xgboost (C) K-nearest neighbor

(D) Decision tree (E) Support vector (F) Neural network

FIGURE 5.25: Prediction and anomaly detection for 6 machine learning algorithms between 3595 ft to 4097 ft Well 58-32

As a result, *random forest*, *xgboost* and *decision tree* identified the same anomalies. However, the results were imprecise to determine what was the correct model in unsupervised learning. In Table 5.12 is presented the performance metric for each of the models related to prediction accuracy. As a aforementioned, metrics of detection were not included due to unlabeled data. Further research will require this information.

In the section of analysis, 3595 ft to 4097 ft in Well 58-32, we can observe that *decision tree* and *support vector regression* obtained the best performance (lower $RMSE$). This scenario of anomaly identification is highly dependent of how precise was the prediction, otherwise this methodology will generate false positives or false negatives

TABLE 5.12: RMSE and run time (sec) for interval between 3595 ft to 4097 ft Well 58-32

Method	**RMSE (ft/hr)**	**Run time (sec)**
Random forest	6,42	0,10
XGBoost	6,30	0,13
K-Nearest neighbor	5,85	0,04
Decision tree	5,73	0,03
Support vector	5,77	0,06
Neural networks	8,67	0,78

alarms. With these assessment, we have finished the experimental result section. In the next section you will find the discussion for prediction and anomaly detection.

Highlights:

- In *stationary analysis*, *random forest* produced better performance in the scenario of random data analysis. In the case of of *incremental data*, again *random forest* endorse its capacity. However, once we divided the data set by formations, different models obtained low error.

- In the case of anomaly detection, machine learning models identified misbehaviour in the case of complete data-set for *stationary analysis*. However, their root cause was not determined. Assessing the distance of the data regard to the prediction is another alternative to identify anomalies. This last strategy requires high accuracy of the machine learning model selected.

Chapter 6

Discussion

6.1 Discussion about prediction and anomaly detection

In general, this research allows us to explore different machine learning algorithms for prediction and anomaly detection. This analysis indicated that machine learning models can provide highly accurate results by applying the proposed methodology with high flexibility. The performance of prediction assessment was more robust having two data set to explore the different alternatives. Meanwhile, the exploration of anomalies were challenging because the data set of both wells (Well 21-31 and Well 58-32) had not identified which points were anomalous. This situation allowed us to explore and research more alternatives to improve the identification of these misbehaviour.

Related to **rate of penetration prediction**, we have found the followings insights:

- Testing with real-world data sets seems to show that more complex algorithms do not always yield the performance gains that one would expect from their sophistication. In our case, *random forest* and *xgboost* obtained better performance than *neural networks* in stationary evaluation.
- Related to run time, *neural networks* achieved the highest value. This scenario decreases the possibilities to implement the algorithm in further analysis. However, even with low accuracy of *neural networks* and high run time, this method has to be in the list of analysis for additional research. Reinforcement learning using *neural networks* has been implemented in others fields. Alternative that could be implemented in drilling operation. In addition, the new developments in deep learning could increase the capacity of the models already evaluated. For instances, with long short term memory (LSTM) neural networks and convolutional neural networks or more complex architectures.
- In hyperparameter optimization, the definition of parameters was defined with specific sequence of values. Recently, researchers suggest to develop the analysis with random values.
- Decreasing the number of features decreased the performance of the predictions. In fact, Soares [58] suggested that could be interesting to evaluate these algorithms considering additional data sources available in real time like gamma ray logs, sonic travel times, bottom-hole pressure and bit torque.
- Increasing the data available to train enabled us to approximate the assessment toward an approximation to real-time scenario. Therefore, this architecture could contribute for further research.
- Exploring predictions per formations with batch assessment granted us to identify that there are not a unique machine learning algorithm with the best capacity. Which represents a different result obtained by Soares [58] where *random*

forest obtained the best performance. In addition, an increasing in the error metric was observed once we were analysing the last batches. It means, a real-time implementation could require a simultaneously assessment to define which model represents the interested patter with better accuracy.

- In the analysis per formation still there is a gap to cover with hyperparameter optimization which was not included in this architecture.
- The same architecture proposed in Chapter 4 which represented a generalization of the implemented code could be useful to develop in another oil & gas field.
- The lithological formation identified for Well 58-32 and Well 21-31 allowed us to delimited the top of each data set for batch analysis. However, geological analysis or interpretation is not included in the present research.
- All the algorithms and analysis were developed in Google Colaboratory. Tools like Azure from Microsoft or AWS (Amazon Web Service) were not included in the development. However, this services reduce time consuming in the development of the solutions and probably would bring the flexibility to execute more experiments. In term of speed, Google Colaboratory offers access to GPU (Graphic Processing Unit) speeding up our analysis.

In the case of **anomaly detection**, we can identify the following characteristic of our analysis:

- The alternatives evaluated in stationary analysis achieved to identify outliers based on the interaction of the different features. However, we still need to research how to identify the root cause of misbehaviour.
- The analysis was developed under unsupervised learning since there were not examples of specific anomalies. Future research will require a data set with anomalies. This scenario will allow to establish performance metrics. Oil & gas companies have the challenge to increase the data structured with anomaly labels which will bring multiple benefits to optimize and increase the accuracy performance of misbehavior detection.
- In stationary analysis, the implementation of each algorithm contained different architecture, definition of parameters and subsequent adjustment. It represented a challenge during the research. At the same time, implementing this solution in streaming data will require high customization of the models.
- Combining the analysis of rate of penetration prediction and anomaly detection, *drilling engineers* can use this predictions as a reference identifying drilling problems. Initially, assessment of machine learning models could be implemented during planning stage and post-drilling evaluation.
- Definition of thresholds were defined arbitrarily. The study of multiple data set will increase the accuracy of proposed architectures.
- As was previously mentioned, deep learning models has been surpassing the performance of traditional machine learning algorithms. Exploring complex architectures will bring new alternatives in misbehavior detection for drilling operation.

To sum up, we are proposing a virtual detector in drilling operation able to optimize the operation and reduce uncertainty in decision making.

6.2 Proposal of virtual detector

Industry is finding multiples options to optimize the process based on data management. Recently, companies have developed strategies from bottom to top to administrate the data flow. From digitization technologies to data science teams have been developed to discover solutions hidden in all the data collected. In parallel, different models or layers have been proposed to implement digital programs. In this case, based on the results of machine learning model, we are proposing a virtual detector constructed with the following layers: *streaming data*, *historical data*, *machine learning detector*, *report alert and optimization*, and *control parameters*. This architecture is focused on data analysis as a source of decision making described in Figure 6.1.

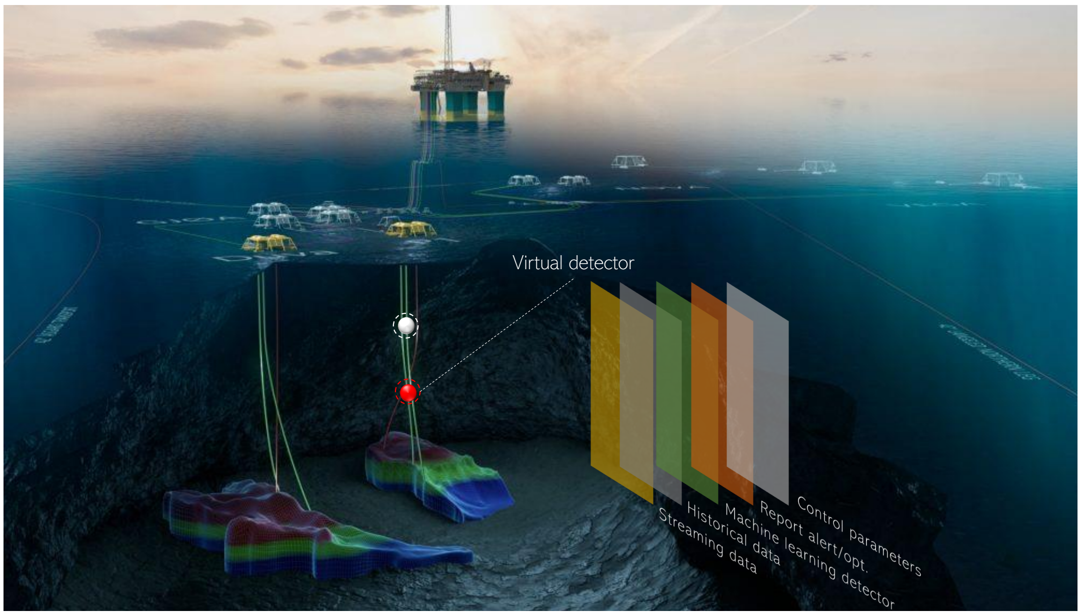
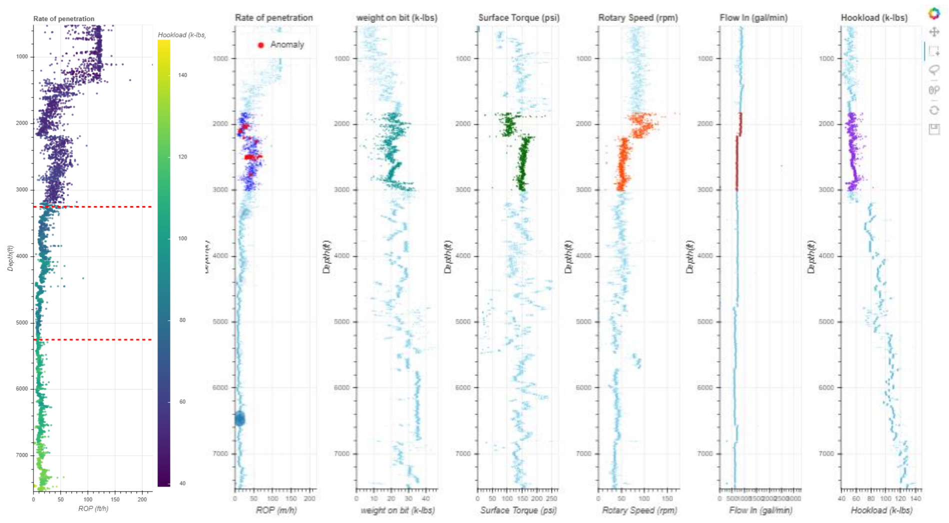


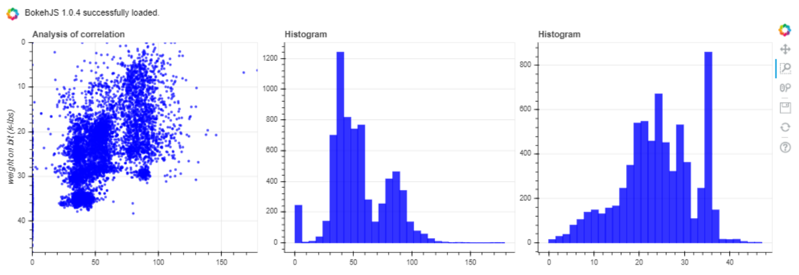
FIGURE 6.1: Proposal of *virtual detector* and respective layers. Image adapted from World Oil [66], Duva oil and gas field

In drilling operation, there are multiple parameters to follow up. As aforementioned in the introduction, to recognize abnormal state and normal state using raw data is difficult for human [53]. Considering this *virtual detector*, we will focus on any algorithm that allow us to reduce the decision uncertainties based on streaming data analysis. Therefore, combining the alternatives evaluated in this research in **machine learning detector layer**, we will improve the process in *real-time operational center*. Assessing the models for anomaly detection, the engineers team will receive alerts of abnormal events that will require attention, as were displayed in Chapter 5. In our case, interactive visualization implemented with *Bokeh* allows to identify the point of anomaly and identify which variable has a deviation. Immediate action can be taken in **control parameter layer** once the flag has be visualized in **report alert layer**. The open source visualization tool implemented in python offered us the option of zoom in and out, get the label of the point with technical information, select one area of interest reflected in the rest of the variables, hide points specially when we are analyzing train and test. In Figure 6.2, we exhibited one of the results of our *virtual detector*.

In addition, **historical data layer** will collect all the results for multiple operation in the cluster. This data becomes the brain of the detector. In addition,



(A) Detection of anomaly



(B) Visualization of data analysis of features

FIGURE 6.2: Report alert and optimization layer in *virtual detector*

this layer will support the planning of drilling engineers. Post analysis of the data using again the *machine leaning detector* on stationary mode will contribute to define which parameters require modification in the respective formation for the next well. In terms of **streaming data**, it is known that the format of data collected is WITSML which stand for wellsite information transfer standard markup language. To sum up, each layer was described in Figure 6.3.

6.3 Personal experience during the project

During the master thesis project different abilities were developed. The implementation of machine learning algorithms allowed me to explore from the statistical theory to coding algorithms. In addition, visualization requirements pushed me to developed better codification skills. In general, all the concept and theory brought me outside my comfort zone. *Patter recognition course* and additional courses in *Coursera* contributed to achieve this result.

The topic called my attention since my participation in *Data bootcamp* in *Rice University (Houston, USA)*. Multiple concepts to learn in a field where there are new advances every day. To keep the track and be updated represent a continuously learning challenge. However, information has always been available. The network in the data science field has been opened with multiple sources about theory and codification. This condition has helped to understand complex and advance models during the implementation.

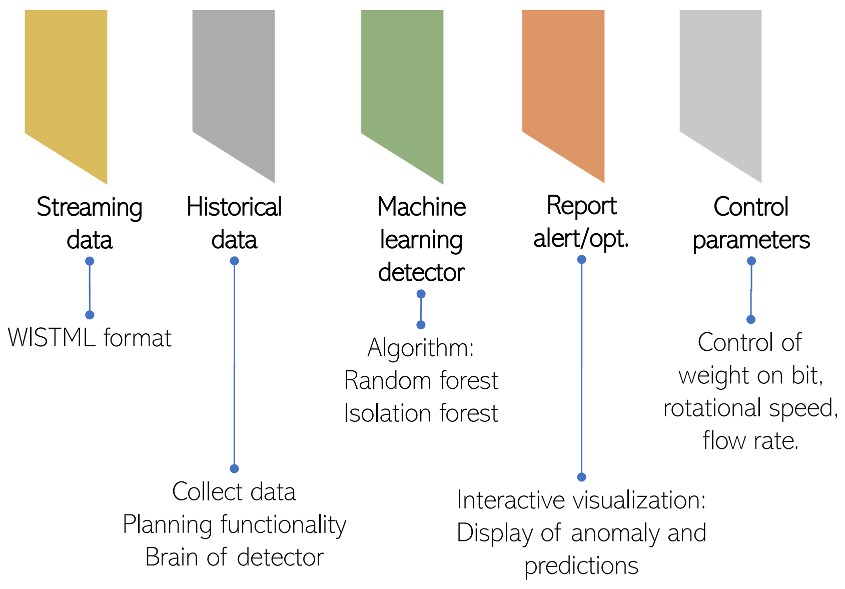


FIGURE 6.3: Layers of *virtual detector*

For my professional path, this research will bring the opportunity to participate in industry digitization programs. Having knowledge of machine learning models will contribute to propose new application and solutions for industrial challenges.

Chapter 7

Conclusion and future work

7.1 Conclusion

This research has presented the evaluation of machine learning models for prediction of respond parameters and anomaly detection in drilling operation. The results demonstrated how the machine learning algorithms could learn continuously in real time operations with accuracy. Based on the proposed objectives we are going to describe our conclusions.

1. *Characterization of machine learning model with stationary analysis and incremental training data available to predict rate of penetration*:
    - *Identify model with best performance (RMSE and run time)*: Based on the data set explored, in the case of stationary analysis, *random forest* and *xgboost* obtained the best performance. Our experiments showed that random forest is a competent machine learning algorithm to predict the rate of penetration with a performance error (root mean squared error) of 2,92 m/hr (9,57 ft/hr) in static analysis and 4,43 m/hr (14,55 ft/hr) average error increasing the availability of data per batches. Testing with real-world data sets seems to show that more complex algorithms like *neural networks* do not always yield the performance gains that one would expect from their sophistication.
    - *Assess incremental data availability strategy using batches per formation (RMSE and run time)* and *analyze training to test set ratio (% training/test)*: Comparing the evaluation increasing the availability of data per batches, we used normalized root mean square error as a metric. In the experiment without limit of formation, we obtained an average error of 0,59 with the best model (*random forest*). However, when we assessed per formation the average error decreased to 0,42 (with *extreme gradient boosting)* for formation 1 and 0,35 for formation 3 (with *k-nearest neighbor*). In the scenario of batches per formation, there are not a unique machine learning algorithm with the best capacity. It means that a real-time implementation will require a simultaneously assessment to define which model represents the interested patter with better accuracy. In terms of time, *neural networks* was the model with major run average time (2.98 sec) compared to *random forest* with 0.35 sec.
    - *Select optimal hyperparameter combination (hyperparameters)*: Cross validation analysis allowed us to identify the best hyper-parameter for each model. Except *neural network*, all the models improved the performance after optimization of parameters. In the case of *neural network* was observed overfitting after the identification of best hyperparameters.

- *Determine relevant features (features with 95 % of importance)*: Reducing the number of features increased the error of the predictions, therefore all the features were used during the experimentation. With 16 features we obtained a root mean square error of 9,99 ft/hr, while with 5 features the root mean square error increased to 10,23 ft/hr.

2. *Identification of misbehavior of operational parameters (control, uncontrolled and response parameters)*: The experiments were executed in unsupervised learning. No labels were available defining normal or abnormal parameters. Therefore, an exploratory analysis was executed in this section. The assessment were implemented in a stationary analysis and applying time series assessment.
   - *Test machine learning methods to identify anomalies in stationary analysis (number of anomaly detection):* Among the machine learning models studied for anomaly detection in stationary analysis, *isolation forest* demonstrated to be a flexible alternative being free of parameters. The methods detected between 52 and 69 anomalies over 6511 points. However, there is no single detector excelling on all type of problems, as different detectors are suitable for different types of problems.
   - *Test machine learning methods to identify anomalies in streaming data (number of anomaly detection):* In the case of time-series, this strategy is highly depend on the accuracy of predictions. However, this section requires further research with labeled data to define the accuracy of the models based on confusion matrix.

Finally, a *virtual detector* was proposed based on this implementation. *Streaming data*, *historical data*, *machine learning detector*, *report alert and optimization*, and *control parameters* are the five layers that constitute the architecture. *Drilling engineers* can use these predictions as a reference for post analysis, optimization and real-time detection of misbehavior. An interactive visualization in *report alert and optimization layer* contributed to improve the data analysis.

7.2 Future work

- *How can drilling operation take advantage of reinforcement learning to improve the autonomy and online learning with respect to a prior unknown system and environment, dynamic uncertainties, and partial observability?.* For this proposal, neural networks will have a relevant functionality. Therefore, better configuration of neural networks architecture will require a deep analysis.

- Companies are requiring to implement solution on real time. Next step will need to adequate the models designed in this research using solutions as Amazon Web Service [6] or Azure [44] to develop experiments with streaming data.

- Having access to a cluster well data set. For instance, if we can get access to drilling data of a number of wells in a determined area under similar conditions of formations. It will allow to identify similar patters of parameters. Under this condition, we could improve the detection of anomalies generating a virtual detector.

- Experiments using deep learning are constantly growing. Therefore, for future research is recommended to evaluate combination of neural networks techniques to improve the detection of misbehavior. For instance, we could link *autoencoder* with *long short term memory* neural networks.

- With more details of the wells, we could include more features like: uncofined compressive strength (UCS) of the different rock types and mud viscosity of mud drilling. At the same time, we could add categorical parameters like bit size, bit type, bit wear, drilling mud type, formation abrasiveness and formation drillability. Uncofined compressive strength, formation abrasiveness and formation drillability could be calculated from lab cores and well logging data. The research developed by Shi et al. [56] used the mentioned features in their neural networks model obtaining high accuracy in their test.

- Currently, oil & gas companies are interested to create *digital twin* of the well, looking for closed-loop optimization at the field level. Therefore, high performance algorithms like machine learning models are improving this solution. The models analyzed in this research could be implemented in a digital twin pilot project.

Bibliography

[1] Martín Abadi et al. “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems”. In: *CoRR* (2016), pp. 1–19.

[2] K.L. Adair. “Extracting knowledge from temporal clusters for real-time clustering”. In: *International Joint Conference on Neural Networks*. Washington, DC, USA: Institute of Electrical and Electronics Engineers (IEEE), July 1999, pp. 2580–2583.

[3] Charu C Aggarwal. *Outlier Analysis*. Yorktown Heights, New York, USA: Springer International Publishing, 2017.

[4] Subutai Ahmad et al. “Unsupervised real-time anomaly detection for streaming data”. In: *Neurocomputing* 262 (Nov. 2017), pp. 134–147.

[5] Kingsley Amadi. “Application of Mechanical Specific Energy Techniques in Reducing Drilling Cost in Deepwater Development”. In: *SPE Deepwater Drilling and Completions Conference*. Galveston, Texas, USA: Society of Petroleum Engineers, 2012.

[6] *Amazon Web Services (AWS) - Cloud Computing Services*. 2019. URL: https://aws.amazon.com/.

[7] Victor Ambonati. *Unsupervised Anomaly Detection | Kaggle*. 2017. URL: https://www.kaggle.com/victorambonati/unsupervised-anomaly-detection.

[8] Riyaz Ahamed Ariyaluran Habeeb et al. “Real-time big data processing for anomaly detection: A Survey”. In: *International Journal of Information Management* 45 (Apr. 2019), pp. 289–307.

[9] Islam Al-Baiyat and Lloyd Heinze. “Implementing Artificial Neural Networks and Support Vector Machines in Stuck Pipe Prediction”. In: *SPE Kuwait International Petroleum Conference and Exhibition*. Kuwait City, Kuwait: Society of Petroleum Engineers, 2012, pp. 1–13.

[10] Ozgur Balamir et al. “Utah FORGE Reservoir: Drilling Results of Deep Characterization and Monitoring Well 58-32”. In: *Workshop on Geothermal Reservoir Engineering*. Stanford, California, USA: Stanford University, 2018, pp. 1–7.

[11] Gerard Biau. “Analysis of a Random Forests Model”. In: *Machine Learning Research* 13 (2012), pp. 1063–1095.

[12] Christopher M Bishop. *Neural Networks for Pattern Recognition*. New York, NY, US: Oxford University Press, Inc., 1995.

[13] Bokeh Development Team. *Bokeh: Python library for interactive visualization*. 2018. URL: https://bokeh.pydata.org/en/latest/.

[14] Leo Breiman. “Random Forests”. In: *Machine Learning* 45 (2001), pp. 5–32.

[15] Leo Breiman. “Technical Note: Some Properties of Splitting Criteria”. In: *Machine Learning* 24 (1996), pp. 41–47.

[16] Jason Brownlee. *How To Backtest Machine Learning Models for Time Series Forecasting*. 2016. URL: https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/.

[17] Jason Brownlee. *How to Develop Machine Learning Models for Multivariate Multi-Step Air Pollution Time Series Forecasting*. 2018. URL: https://machinelearningmastery.com/how-to-develop-machine-learning-models-for-multivariate-multi-step-air-pollution-time-series-forecasting/.

[18] Jason Brownlee. *Multivariate Time Series Forecasting with LSTMs in Keras*. 2017. URL: https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/.

[19] Kalle Burbeck and Simin Nadjm-Tehrani. "ADWICE-Anomaly Detection with Real-time Incremental Clustering". In: *International Conference on Information Security and Cryptology*. Berlin, Heidelberg: Springer, 2004, pp. 407–424.

[20] Varun Chandola. "Anomaly Detection : A Survey". In: *ACM Computing Surveys (CSUR)* 41.3 (2009), 15:1–15:58.

[21] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *CoRR* (2016).

[22] François Chollet. *Keras*. 2015. URL: https://keras.io/.

[23] EGI. *Frontier Observatory for Research in Geothermal Energy Phase 2B*. Tech. rep. Utah: Utah Forge, 2018.

[24] Junliang Fan et al. "Comparison of Support Vector Machine and Extreme Gradient Boosting for predicting daily global solar radiation using temperature and precipitation in humid subtropical climates: A case study in China". In: *Energy Conversion and Management* 164 (May 2018), pp. 102–111.

[25] Maurizio Filippone et al. "A survey of kernel and spectral methods for clustering". In: *Pattern Recognition* 41.1 (2008), pp. 176–190.

[26] Roni Gandelman et al. "Field Implementation of a Real Time Drilling Problem Diagnostic for Deepwater Exploratory Wells". In: *Offshore Technology Conference*. Houston, Texas, USA: Offshore Technology Conference, 2010, pp. 3–6.

[27] John-Morten Godhavn et al. "Drilling seeking automatic control solutions". In: *World Congress The International Federation of Automatic Control*. Vol. 44. 1. Milano, Italy: IFAC, Jan. 2011, pp. 10842–10850.

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Cambridge: The MIT Press, 2017.

[29] Chiranth Hegde and K E Gray. "Use of machine learning and data analytics to increase drilling efficiency for nearby wells". In: *Natural Gas Science and Engineering* 40 (2017), pp. 327–335.

[30] Chiranth Hegde and Ken Gray. "Evaluation of coupled machine learning models for drilling optimization". In: *Journal of Natural Gas Science and Engineering* 56 (2018), pp. 397–407.

[31] Chiranth Hegde, Cesar Soares, and Ken Gray. "Rate of penetration (ROP) modeling using hybrid models: deterministic and machine learning". In: *Unconventional Resources Technology Conference*. Houston, Texas, USA: Unconventional Resources Technology, 2018, pp. 23–25.

[32] Chiranth Hegde, Scott Wallace, and Ken Gray. "Real Time Prediction and Classification of Torque and Drag During Drilling Using Statistical Learning Methods". In: *SPE Eastern Regional Meeting*. Morgantown, West Virginia, USA, Society of Petroleum Engineers, 2015, pp. 13–15.

[33] Chiranth Hegde, Scott Wallace, and Ken Gray. "Using Trees, Bagging, and Random Forests to Predict Rate of Penetration During Drilling". In: *SPE Middle East Intelligent Oil & Gas Conference & Exhibition*. Abu Dhabi, UAE: Society of Petroleum Engineers, 2015, pp. 1–12.

[34] Chiranth Hegde et al. "Analysis of rate of penetration (ROP) prediction in drilling using physics-based and data-driven models". In: *Journal of Petroleum Science and Engineering* 159 (Nov. 2017), pp. 295–306.

[35] Helmi Helmiriawan. *Scalability Analysis of Predictive Maintenance Using Machine Learning in Oil Refineries*. Tech. rep. Delft University of Technology, 2018.

[36] Robin Hes. *Insurance A Machine Learning Perspective*. Tech. rep. Delft University of Technology, 2018.

[37] John D. Hunter. "Matplotlib: A 2D Graphics Environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.

[38] Gareth James et al. *An Introduction to Statistical Learning with applications in R*. New York: Springer, 2003.

[39] Will Koehrsen. *Data Visualization with Bokeh in Python, Part I: Getting Started*. 2018. URL: https://towardsdatascience.com/data-visualization-with-bokeh-in-python-part-one-getting-started-a11655a467d4.

[40] Kurt O Kraal and Bridget Ayling. "Hyperspectral Characterization of Fallon FORGE Well 21-31: New Data and Technology Applications". In: *Workshop on Geothermal Reservoir Engineering*. California: Stanford University, 2019, pp. 1–15.

[41] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation-based Anomaly Detection". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6.1 (2012), 3:1–3:39.

[42] Sankar Mahadevan and Sirish Shah. "Fault detection and diagnosis in process data using one-class support vector machines". In: *Process Control* 19 (2009), pp. 1627–1639.

[43] Wes Mckinney. "Data Structures for Statistical Computing in Python". In: *9th Python in science conference*. Scipy, 2010, p. 51.

[44] *Microsoft Azure Cloud Computing Platform*. 2019. URL: https://azure.microsoft.com/en-us/.

[45] Gopinath Muruti, Fiza Abdul Rahim, and Zul-Azri bin Ibrahim. "A Survey on Anomalies Detection Techniques and Measurement Methods". In: *IEEE Conference on Applications, Information and Network S ecurity (AINS)*. Malaysia: IEEE Computer society, 2018, pp. 81–86.

[46] OpenEI. *Geothermal Data Repository (GDR)*. 2018. URL: https://gdr.openei.org/submissions/1113.

[47] F. Pedregosa. *Cross-validation: evaluating estimator performance*. 2012. URL: https://scikit-learn.org/stable/modules/cross_validation.html.

[48] Fabian Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Machine Learning Research* 12 (Nov. 2011).

[49] Tomáš Pevný and Joao Gama B Tomáš. “Loda: Lightweight on-line detector of anomalies”. In: *Machine Learning* 102 (2016), pp. 275–304.

[50] Jacob Pollock, Zachary Stoecker-Sylvia, and Vinod Veedu. “Machine Learning for Improved Directional Drilling”. In: *Offshore Technology Conference*. Houston, Texas, USA: Offshore Technology Conference, 2018, pp. 1–9.

[51] J R Quinlan. “Induction of Decision Trees”. In: *Machine Learning* 1 (1986), pp. 81–106.

[52] Markus Reichstein. “Deep learning and process understanding for data-driven Earth system science”. In: *Nature* 566 (2019), pp. 195–204.

[53] Mayu Sakurada and Takehisa Yairi. “Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction”. In: *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis - MLSDA’14*. Gold Coast, Australia: ACM Press, 2014, pp. 4–11.

[54] Sandia National Laboratories. *Fallon, Nevada FORGE Lithology Logs and Well 21-31 Drilling Data - Data.gov*. 2018. URL: https://catalog.data.gov/dataset/fallon-nevada-lithology-logs-and-well-21-31-drilling-data.

[55] Bernhard Scholkopf et al. “Estimating the Support of a High-Dimensional Distribution”. In: *Neural Computation* 13 (2001), pp. 1443–1471.

[56] Xian Shi et al. “An Efficient Approach for Real-Time Prediction of Rate of Penetration in Offshore Drilling”. In: *Mathematical Problems in Engineering* 2016 (Nov. 2016), pp. 1–13.

[57] Alex J Smola and Bernhard Scholkopf. “A tutorial on support vector regression”. In: *Statistics and Computing* 14 (2004), pp. 199–222.

[58] Cesar Soares and Kenneth Gray. “Real-time predictive capabilities of analytical and machine learning rate of penetration (ROP) models”. In: *Petroleum Science and Engineering* 172 (2019), pp. 934–959.

[59] James Speight. *Handbook of Offshore Oil and Gas Operations Drilling Technology and Well Completion*. Elsevier Ltd., 2011, pp. 127–148.

[60] Swee Chuan Tan, Kai Ming Ting, and Tony Fei Liu. “Fast Anomaly Detection for Streaming Data”. In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*. Barcelona: AAAI Press, 2011, pp. 1511–1516.

[61] Cenk Temizel et al. “Turning Data into Knowledge: Data-Driven Surveillance and Optimization in Mature Fields”. In: *SPE Annual Technical Conference and Exhibition*. Vol. 2. Dubai, UAE: Society of Petroleum Engineers, 2016, pp. 1–32.

[62] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern recognition*. London, UK: Elsevier, 2009.

[63] Sean Unrau. “Machine Learning Algorithms Applied to Detection of Well Control Events”. In: *SPE Kingdom of Saudi Arabia Annual Technical Symposium and Exhibition*. Dammam, Saudi Arabia: Society of Petroleum Engineers, 2017, pp. 1–10.

[64] Vladimir N Vapnik. “An Overview of Statistical Learning Theory”. In: *Transactions on Neural Networks* 10.5 (1999), pp. 988–999.

[65] Scott P Wallace, Chiranth M Hegde, and K E Gray. “A System for Real-Time Drilling Performance Optimization and Automation Based on Statistical Learning Methods”. In: *SPE Middle East Intelligent Oil & Gas Conference & Exhibition*. Abu Dhabi, UAE: Society of Petroleum Engineers, 2015, pp. 1–12.

[66] World Oil. *Neptune Energy, TechnipFMC enter into subsea global alliance agreement*. 2019. URL: https://www.worldoil.com/news/2019/4/3/neptune-energy-technipfmc-enter-into-subsea-global-alliance-agreement.

[67] Jie Zhao et al. “Machine Learning-Based Trigger Detection of Drilling Events Based on Drilling Data”. In: *SPE Eastern Reg ional Meeting*. Lexington, Kentucky, USA: Society of Petroleum Engineers, 2017, pp. 4–6.

Appendix A

Performance of rate of penetration prediction per batches

A.1 Results of rate of penetration prediction per batches Well 58-32

The performance results of predictions per batches for each machine learning model are presented in the following tables. First, the outcome of increasing data available per batch without limit of formation is presented in Table A.1, Table A.2, and Table A.3. In the second part, we find the results of increasing data available per batch per formation in Table A.4, Table A.5 and Table A.6. These information is support for the analysis in Chapter 5.

TABLE A.1: Root mean square error (ft/hr) per batch increasing data available Well 58-32

BATCH	**LR**	**KNR**	**SVR**	**DT**	**RF**	**XGB**	**ADABR**	**MLP**
1	103.17	69.40	64.56	38.86	44.18	51.80	48.82	38.86
2	34.50	19.77	32.01	28.32	21.35	18.97	24.67	28.32
3	37.40	15.45	16.35	21.90	16.79	19.19	16.25	21.90
4	17.81	13.89	17.43	25.30	19.00	20.54	22.29	25.30
5	23.73	16.20	30.15	9.49	8.55	14.81	29.49	9.49
6	29.70	3.31	7.83	11.61	4.04	5.62	29.32	11.61
7	24.91	4.28	7.49	6.95	5.17	3.28	19.77	6.95
8	32.99	5.83	6.01	8.65	5.85	4.91	17.46	8.65
9	24.31	6.83	6.26	7.91	6.02	6.13	9.93	7.91

Table A.2: Normalized root mean square error per batch increasing data available Well 58-32

BATCH	**LR**	**KNR**	**SVR**	**DT**	**RF**	**XGB**	**ADABR**	**MLP**
1	1.98	1.33	1.24	0.75	0.85	1.00	0.94	0.75
2	0.92	0.52	0.85	0.75	0.57	0.50	0.65	0.75
3	1.02	0.42	0.44	0.60	0.46	0.52	0.44	0.60
4	1.02	0.80	1.00	1.45	1.09	1.18	1.28	1.45
5	1.62	1.10	2.05	0.65	0.58	1.01	2.01	0.65
6	3.36	0.37	0.89	1.31	0.46	0.64	3.31	1.31
7	2.47	0.42	0.74	0.69	0.51	0.32	1.96	0.69
8	2.50	0.44	0.46	0.66	0.44	0.37	1.33	0.66
9	1.65	0.46	0.43	0.54	0.41	0.42	0.67	0.54

Table A.3: Time (sec) per batch increasing data available Well 58-32

BATCH	**LR**	**KNR**	**SVR**	**DT**	**RF**	**XGB**	**ADABR**	**MLP**
1	0.00	0.02	0.07	0.02	0.07	0.08	0.13	0.85
2	0.00	0.04	0.16	0.02	0.12	0.14	0.20	1.50
3	0.00	0.03	0.30	0.03	0.18	0.20	0.25	2.26
4	0.00	0.04	0.50	0.04	0.25	0.25	0.30	2.94
5	0.00	0.10	0.68	0.05	0.31	0.31	0.36	3.29
6	0.00	0.05	0.86	0.06	0.39	0.39	0.45	3.92
7	0.00	0.06	1.04	0.07	0.45	0.42	0.50	2.57
8	0.00	0.07	1.24	0.09	0.54	0.47	0.57	4.07
9	0.00	0.08	1.52	0.10	0.62	0.52	0.61	5.47

Table A.4: Normalized root mean square error per batch increasing data available in *F1* Well 58-32

BATCH	**LR**	**KNR**	**SVR**	**DT**	**RF**	**XGB**	**ADABR**	**MLP**
1	0.41	0.14	0.14	0.43	0.30	0.32	0.24	0.43
2	0.27	0.25	0.26	0.32	0.27	0.29	0.25	0.32
3	1.61	0.84	1.00	0.69	0.68	0.61	0.75	0.69
4	0.50	0.53	0.64	0.50	0.51	0.40	0.85	0.50
5	0.63	0.65	1.11	1.68	0.98	0.64	1.29	1.68
6	1.11	0.50	0.73	0.68	0.54	0.51	0.52	0.68
7	0.83	0.26	0.49	0.39	0.30	0.28	0.26	0.39
8	0.31	0.34	0.32	0.44	0.34	0.35	0.34	0.44
9	1.17	0.42	0.36	0.45	0.36	0.36	0.42	0.45

Table A.5: Normalized root mean square error per batch increasing data available in *F2* Well 58-32

BATCH	**LR**	**KNR**	**SVR**	**DT**	**RF**	**XGB**	**ADABR**	**MLP**
1	1.25	1.28	1.39	1.85	1.65	1.82	1.69	1.85
2	0.62	1.05	1.18	0.90	1.22	1.20	1.28	0.90
3	0.54	0.82	0.83	1.92	1.01	0.92	1.06	1.92
4	1.26	1.06	0.83	0.73	0.95	0.74	0.96	0.73
5	1.54	1.60	1.45	1.68	1.65	1.54	1.77	1.68
6	1.20	1.01	1.44	1.98	1.72	2.12	2.80	1.98
7	0.59	1.33	1.33	1.45	1.37	1.68	1.54	1.45
8	0.80	1.04	1.13	1.32	0.99	1.09	1.45	1.32
9	1.54	1.27	1.41	1.52	1.36	1.49	1.65	1.52

Table A.6: Normalized root mean square error per batch increasing data available in *F3* Well 58-32

BATCH	**LR**	**KNR**	**SVR**	**DT**	**RF**	**XGB**	**ADABR**	**MLP**
1	0.48	0.33	0.30	0.40	0.30	0.29	0.28	0.40
2	0.61	0.34	0.30	0.39	0.37	0.36	0.28	0.39
3	0.31	0.31	0.37	0.72	0.53	0.49	0.44	0.72
4	0.47	0.53	0.48	0.42	0.39	0.36	0.36	0.42
5	0.35	0.31	0.34	0.51	0.45	0.38	0.37	0.51
6	0.39	0.41	0.39	0.54	0.47	0.49	0.52	0.54
7	0.50	0.33	0.27	0.34	0.29	0.32	0.28	0.34
8	0.30	0.28	0.36	0.43	0.34	0.33	0.32	0.43
9	0.48	0.39	0.49	0.50	0.47	0.48	0.48	0.50