University of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialization: MSc. Petroleum Engineering (Drilling and Wells) Engineering	Spring semester, 2022 Open Source
Author: Miguel Fernández Berrocal	[illegible] .... Miguel Fernández Berrocal
Internal Supervisor: UiS - Prof. Alf Co-Supervisor: UiS - Prof. Dan Sui External Supervisors SEKAL AS	
Title of master's thesis: ESTIMATION OF THE FREE ROTATING HOOK LOAD FOR WOB ESTIMATION	
Credits: 30	
Keywords: Hook load, Drilling, Optimization, Machine Learning	Number of pages: + supplemental material/other: 77 Stavanger, 15th June 2022

Miguel Fernández Berrocal

ESTIMATION OF THE FREE ROTATING HOOK LOAD FOR WOB ESTIMATION

MsC Project for the degree of
MSc in Petroleum Engineering

Stavanger, June 2022

University of Stavanger
Faculty of Science and Technology
Department of Energy and Petroleum Engineering

Universitetet
i Stavanger

Today's drilling industry emphasizes safety and deeper drilling while reducing drilling costs. Low rate of penetration (ROP) and non-productive time are two main reasons for reduced drilling efficiency.

Machine Learning (ML) technology has been increasingly used in the Oil and Gas industry for a variety of problems, including drilling parameters estimation and prediction, drilling incidents detection, and optimal well planning.

With the abundance of field data available, a number of detailed research studies have been conducted to define the relationship between the ROP and drilling parameters. However, developing a pure data-driven ROP model and optimization remains very challenging. The main reason for this is the number of parameters that affect its estimation, as well as the manner in which different variables are correlated, e.g., the downhole weight on bit (DWOB), rotary speed (RPM), standpipe pressure (SPP), and formation/bit properties [1]. Thus, we suggest breaking down the ROP data-driven problem into its primary parameters and focusing on them separately.

Analyzing time series for hook load when drilling with connections is the purpose of our study. In drilling operations, hook load is used to control the weight on the bit. Due to the inability to measure *WOB* directly during drilling, the tension at the top of the drill string is used to determine *WOB*. As of now, the industry is letting the driller select manually a free-rotating hookload choice according to his judgment and experience. This task of manual inspection and selection is tedious and, according to experts, often left behind due to the number of tasks at a time the driller has to deal with. This recorded value will be used to calculate the *WOB* indirectly while drilling. With an industry that is more autonomous than ever before, this would not be a viable option, and a different approach is proposed in this research.

In essence, this research is an attempt to estimate automatically and more accurately rather than relying on the experience and judgments of the driller, the free rotating hook load. The approach will be accomplished by developing a hybrid system combining ML algorithms with statistical analysis and physics principles.

As a result of embedding this rig state identification engine idea, we will be able to classify and analyze different real-time data points (e.g., out-of-slips, pick-up, rotating off-bottom, and drilling), obtain the free rotating hook load, and also to utilize it in other applications, such as *T&D* calibration.

Acknowledgments

First and foremost, for their supervision, guidance, and for sharing his time and knowledge with me, I would like to thank Associate Professor Alf Kristian Gjerstad and Dan Sui from the University of Stavanger. Also, to the Sekal team to offer me suggestions and new concerns to improve my work. It is also my honor to acknowledge my grandparents, who died before I finished college. They would be proud and I will always be grateful for the priceless memories I have from them. I am grateful to my family for their continuous support and trust, even when they were far away, encouraging and accompanying me throughout all my endeavors. Especial thanks to my love Alina who stuck with me during the long months of writing and rewriting.

Miguel Fernández Berrocal

List of Abbreviations

ADC	Automated drilling control
AI	Artificial Intelligence
BHA	Bottom Hole Assembly
BHP	Bottom hole pressure
CB	Crown block
CS	Control system
CSV	Comma-Separated Value
DWOB	Downhole Weight on Bit
DHTQE	Downhole Torque
DW	Draw-works
ECD	Equivalent Circulating Density
EMS	Enhanced measurement system
FRW	Free rotating weight
GMM	Gaussian Mixture Models
HPHT	High Pressure High Temperature
HKLD	Hook load
ID	Inside Diameter
IQR	Interquartile Range
KNN	K-Nearest Neighbors
LWD	Logging While Drilling
MAE	Mean Absolute Error
ML	Machine Learning
MSE	Mechanical Specific Energy
MWD	Measuring While Drilling
R^2	Coefficient of Determination
OD	Outside Diameter
ROP	Rate of Penetration
RPM	Revolutions per minute
SPP	Standpipe Pressure
STQ	Surface Torque
SWOB	Surface Weight on Bit
TD	Top Drive
WOB	Weight on Bit
WDP	Wired Drill Pipe

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background, Motivation and Challenge

Having a simple and well-defined equation that solves a problem is very convenient. However, we are not always able to construct robust mathematical models to describe the system, which makes a physics-based model impractical and unreliable. This is the case, for instance, for weight on bit calculation.

To acquire better drilling performance, weight on bit *(WOB)* needs to be accurate and monitored. Downhole Weight on bit sensor measurements is usually unavailable or inaccurate. Due to the lack of techniques for investigating and analyzing downhole weight on bit, which may differ dramatically from surface measurements, a gap remains between the planned optimization of a drilling program and its actual implementation. Thus, there is an urgent need for a new technique that allows accurate and reliable surface weight on bit estimation both in real-time for identifying drilling problems and in advance for drilling planning.

Currently, at the start of the drilling process, the bit is typically a few meters above the ground, and the drill string is set to rotate continuously. In *'Rotating Off-Bottom '* mode, the drill string is simply rotated without any movement in the axial direction. In this case, there is no absolute velocity in the vertical direction. Next, to prevent the bit from bouncing, the bit is slowly lowered onto the bottom. Prior to tagging the bottom, the free-rotating hook load is measured and recorded while the drill string is free-rotating. In this way, the maximum and minimum steady-state hook loads are measured. The fluctuations of the hook load are mainly caused by the rotation of the bit and drill string which encounters differing forces as they are rotated. Also contributing to the fluctuations is the flexibility of the pipe, which twists as it rotates.

At present, the driller is given the freedom to choose manually according to his judgment and experience a free-rotating hook load, also known as *'tare hook load'* , when he believes the bit is still both off-bottom and the borehole simulates drilling conditions. On top of that, once the free-rotating hook load is recorded, it remains constant for the whole drilling operation instead of capturing an *off-bottom HKLD* trend that changes over time as drilling proceeds. While drilling, surface *WOB* is indirectly calculated by subtracting the off-bottom *HKLD* from the measured *HKLD* from the sensor. In addition, the *WOB* is also corrected for possible forces affecting *WOB* while drilling, such as hydraulic and mechanical forces. As noticed, it is not an optimal and accurate approach to estimate *WOB* inaccurately due to the low reliability of the free-rotating hook load value, which is a manually selected value by the driller.

As is evident, current industry practices lack the accuracy and efficiency needed to estimate *WOB*, given, among other aspects, the low reliability of its key component: the free-rotating *HKLD* value manually selected by the driller. Hence, our main motivation is to bridge this gap by developing a hybrid model that combines ML, physics, and statistics to catch the Free-rotating hook load *(Off-bottom hook load).* As a result, the drillers can estimate the downhole weight on bit autonomously and accurately, allowing them to make safe and effective decisions.

Thanks to a large amount of available and unused field data, we chose for part of our study statistics as well as Machine Learning as our approach to developing our model. The ML approach does not require a very deep knowledge of physics but rather a good understanding of the learning algorithms and statistics. It deals with the ability of computers/machines to make decisions and act like humans. It is based on learning mode (supervised, unsupervised, and reinforcement learning) as well as it can be categorized as performing the task of classification or regression based on the qualitative or quantitative nature of output data.

In this study, we will specifically focus on unsupervised learning classification for detecting the range at which we start rotating freely until the bit touches the downhole before we may continue drilling. The most common unsupervised machine learning classification algorithms are k-means clustering, isolation forest, and fuzzy c-means. While some machine learning algorithms may be more robust than others in terms of learning accuracy, other factors such as speed of implementation, ease of interpreting results, nature of the machine learning task (classification or regression) and amount of input and output data, as well as the complexity of the model to be learned are important and will be taken into consideration when deciding the unsupervised Machine Learning algorithm to employ. Also, as previously mentioned, domain knowledge is integrated into our research in the form of mathematical and physical equations, as well as logic rules, in order to perform better than purely data-driven models.

1.2 Objectives and Scope

To be able to estimate a more accurate weight on the bit subsequently, this study aims to identify the optimal hook load while free-rotating. It allows the right information to reach the right place at the right time, enabling more informed decisions during critical operations, which will aid in improving drilling performance. By using data-driven models and statistical and physical approaches, the study will offer new insights into the fascinating topic. We propose the following step-by-step objectives to achieve the above-mentioned:

- An understanding of all the components involved in drilling a well is necessary.
- Identify the key factors involved in *WOB* estimation.
- Choose a data set with relevant and consistent information.
- Preparation and cleaning of the data for further analysis.

- Detection of whether we are out-of-slips or in-slips through pattern recognition. That is, differing from the two different operations while drilling operations. One cluster will fall into 'Abnormal operations' (in-slips) and the other into 'Normal operations' (out-of-slips) for every stand.

- Implement an unsupervised Machine Learning classification algorithm to detect clusters. The first cluster corresponds to the range of data from the moment we pick up the drill-string right out-of-slips until we touch bottom. On the other hand, the rest of the clusters comprise the period corresponding to drilling.

- Estimation of the optimal Hook load that resembles the most to drilling operations corresponding to the period we start rotation and pumps are activated until we tag bottom through a statistical and physical constrained approach.

- Detect a trend to be able to anticipate an optimal off-bottom hook load after each new connection to assist the driller during the automated sequence. Also, update *'tare hook load'* over the drilling operations period for a more accurate weight on bit estimation.

Understanding how all the parameters involved have been measured and obtained is paramount to achieving the first four objectives and ensure data generality and quality. As soon as this is done, it is then possible to search and select a suitable data set that contains the required information and key relevant parameters for our research.

We then remove the period of time when we were in-slips from our dataset after identifying the part of the dataset that corresponds to the 'out-of-slips' status for every new stand. In this new dataset, we implement unsupervised Machine Learning to differentiate the clusters of interest. Subsequently, we discuss the model evaluation, interpretation, generality, robustness, and stability in order to overcome one of the barriers to applying data-driven techniques to drilling operations in practice.

Next, the data is narrowed down to only the cluster of main interest, the first cluster. Using a statistical and physical approach to this new dataset, we will determine the period of time we start free-rotating and pumps are activated until we tag bottom. In this narrow time window, we expect to finally obtain the best hook load to continue drilling. Then, the real force applied to the bottom of the well (WOB) will be estimated more accurately for the next ca. 30 meters. The same process is repeated for every new stand that is about to be drilled.

The final step is detecting a trend and predicting accurate future optimal free-rotating hook loads for future connections, as long as the borehole and trajectory conditions do not differ greatly from connection to connection.

1.3 Methodology

A flow chart of the methodology may be found in the figure attached 1.1. For this study, Jupyter Notebook [3] will be the application of choice since it is user-friendly, supports the selected programming language Python [4], and allows to set up an appropriate environment. In Appendix A, you will find a list of the packages used to set the proper programming environment.

For the study to succeed, a comprehensive data set is required. This data set must include, among other things: sensor measurements of the variables, a sufficient quantity of observations, and consistent data along the well trajectory, so as to enable us to run Machine Learning models. It is challenging to obtain such a set of data, which is where data analysis techniques come into play to correct faulty measurements, remove noise, and fill in data gaps. In Chapter 4 we will discuss these techniques.

Once the well data is preprocessed, a Python code [5] based on a statistical analysis called ’Change Point Detection’ is implemented to automatically recognize out-of slips and in-slips statuses, as discussed in Chapter 5.

The next part of the study consists of the evaluation of different Unsupervised Machine Learning Classification techniques with different sets of data. As a result, we will be able to

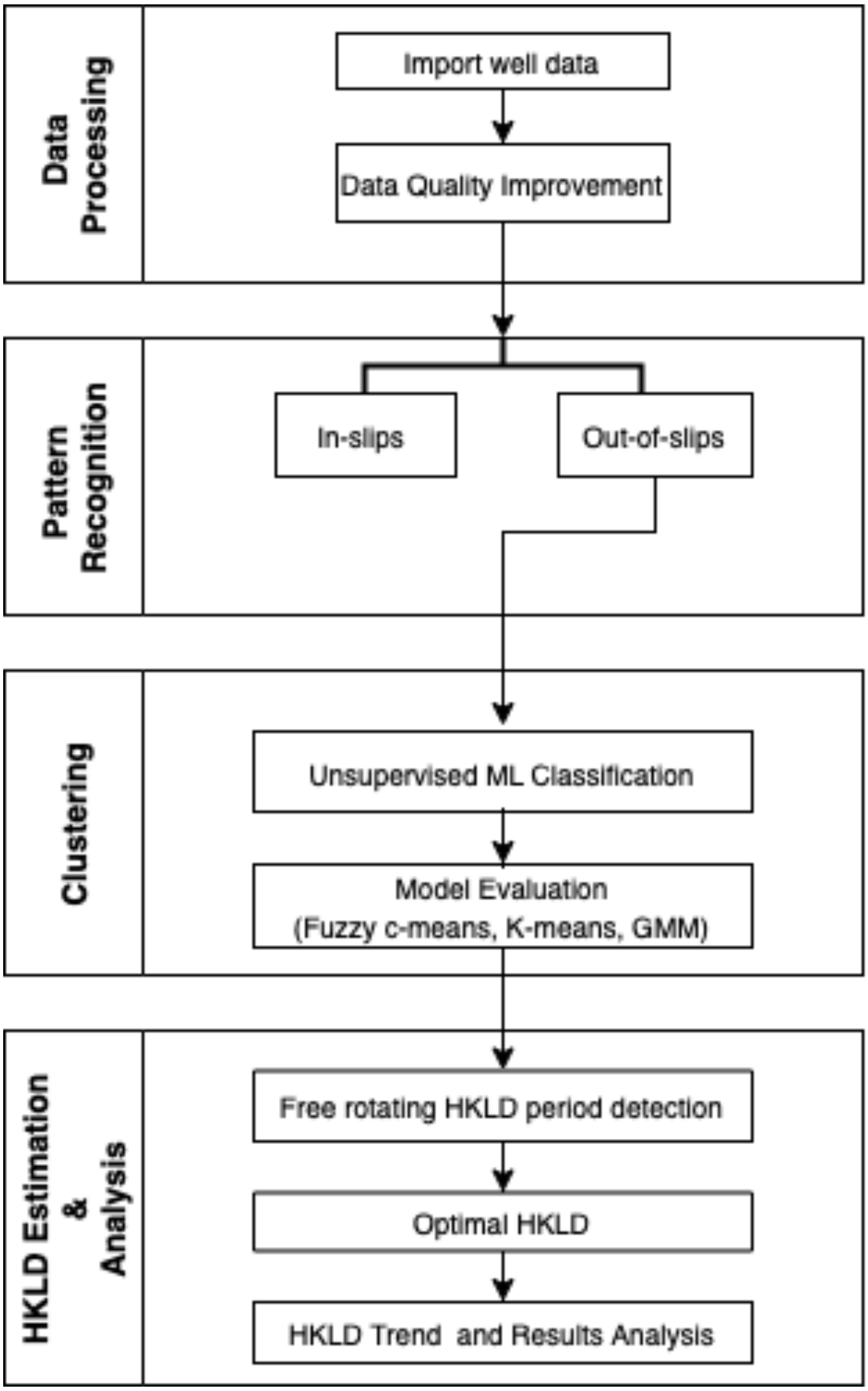

Data Processing
Import well data
Data Quality Improvement
Pattern Recognition
In-slips
Out-of-slips
Clustering
Unsupervised ML Classification
Model Evaluation (Fuzzy c-means, K-means, GMM)
HKLD Estimation & Analysis
Free rotating HKLD period detection
Optimal HKLD
HKLD Trend and Results Analysis

Figure 1.1: Methodology Flow chart

identify the model that is the most accurate in identifying the cluster of interest. Chapter 6 presents the results and evaluations of the ML models.

Based on the ML model that provides the best clustering results, physics and statistics are used to determine the free-rotating hook load time period and to predict future connections. See Chapter 7 for more details.

Chapter 2

Literature Review

During the study, it is essential to first make the reader aware of how surface drilling rig elements work and how they are connected in order to interpret the results. To accomplish this, the first section will be divided into the following subsections:

- Surface Drilling Rig Elements.
- Downhole Drilling Rig Elements.

Secondly, we will focus on offering a clear and brief explanation of the development of downhole data collection and transfer techniques over time, especially as the quality of the measurements have a big impact in our particular study.

Lastly, we will detail a literature review of unsupervised machine learning, where machine learning algorithms are used to identify and group unlabeled data and comment on the diverse methods that were considered for our study.

2.1 Drilling Rig Elements - Surface

The surface elements involved in drilling operations are:

- Hoisting System.

- Rotating Equipment.
- Circulating System.
- Power System.
- Blowout Preventers.
- Mud-Gas Separators.

Our focus will be on the Hoisting, Rotating Equipment, and Circulating System; because the measurements of the parameters involved in them will enable us to determine what is the most optimal tare HKLD. Our research was mainly evaluated using surface measurements since this data is widely available in most drilling rigs.

2.1.1 Hoisting System - Introduction

The Hoisting System supports the weight of the pipe during drilling operations. Depending on the situation, lowering or raising the pipe will be necessary. This system consists of:

- Derrick.
- Traveling Block.
- Drilling Line.
- Crown Block.
- Drawworks.

During the different drilling phase operations, the drill string is suspended from the traveling block, either by elevators or attached to the Top Drive. Several sheaves transmit the weight of the drill string and traveling block to the derrick through the drilling line. The block and tackle scheme described by Bourgoyne et al. [2] clarifies this concept, as shown in the next figure 2.1.

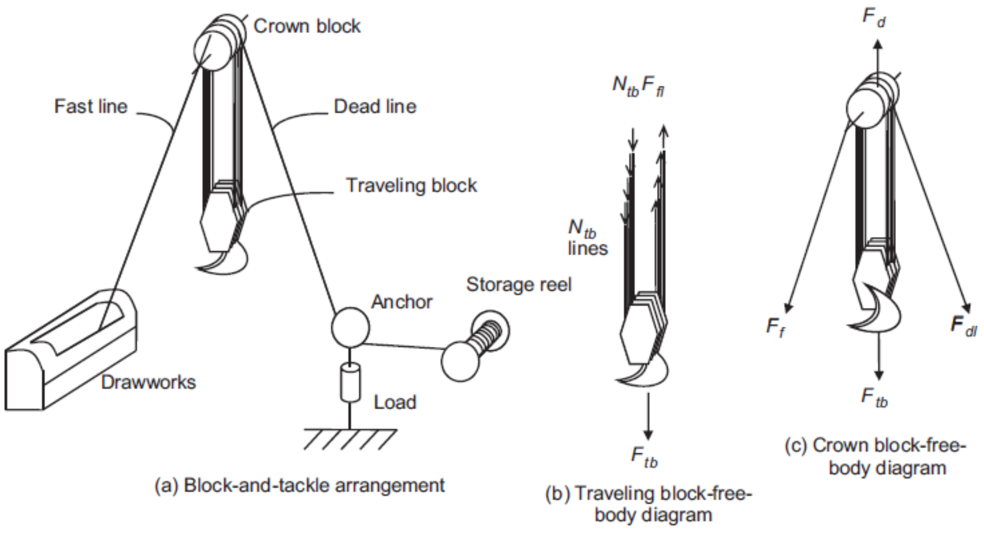


Figure 2.1: Block and Tackle Schematic. Source: [2]

Hoisting System - HKLD

A crucial element of this system is highlighted in figure 2.1, namely the cell load. In the cell, the weight and position (deadline anchor) of the drill string, also known as HKLD, is recorded. In drilling operations, hook load serves as an important parameter used to control the weight on bit as well as to assess possible deteriorations of a hole's condition, such as poor hole cleaning or excessive tortuosity [6].

According to Mme et al. [7], hook load consists of the sum of the forces acting on the drill string attached to the hook. Similarly, in accordance with Cayeux et al. [8], HKLD is determined by forces acting on the drill string. These forces include buoyant weight, mechanical friction, and hydraulic friction forces. Therefore, buoyancy, drag forces produced by friction, fluid shear stress at pipe walls, and inertial forces in curved sections are all factors that affect hook load.

Hoisting System - HKLD Measurement

HKLD is usually measured in practice either in the traveling equipment or in the dead-line tension, as shown in figure [2.2]. It is relevant to know the position of this sensor, as it will include, or not, additional loads in the system. During the measurement process, the apparent hook load is subject to a variety of factors, including the weight of the mud hose attached to the top drive, imperfect tension transmission through sheaves and gravitational, and inertia forces associated with the rotation and weight of the drill line. These forces contribute several metric tons to the hook load, so they need to be accounted for when estimating the true hook load [9].

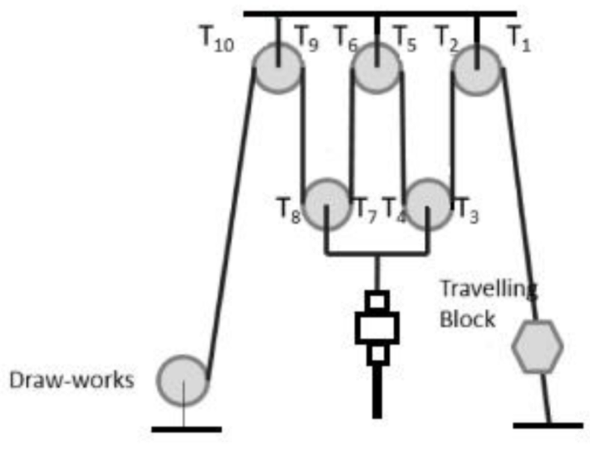


Figure 2.2: Hoisting system

As wells become more complex, sensors are placed closer to the source of measurement. With the aid of special subs like those presented in Hareland et al. [10] work, the true weight of the drill string can be measured. This was connected directly to the drill string in a Top Drive that was able to lodge the tool and measure the weight of the drill string. For the rigs analyzed in this study, the HKLD sensor is placed on the drive sub on top or bottom of the top drive. This means that based on the available data, it is not necessary to consider some of the loads that could distort the real from the recorded weight of the drill string, in this case, the sheave friction effect, weight of the traveling block, and standpipe pressure effect.

Nevertheless, as mentioned before, due to the lack of this capability in most drilling rigs, the hook load is measured indirectly through the deadline or by a load cell placed at the hanging point of the Top Drive. In that case scenario, they will have to take into consideration loads that are not directly related to the weight of the drill string [8].

Moreover, we must take into account the fact that all real-life systems are subject to randomness. More specially, drilling operations present many sources of uncertainty due to bit rock interactions and fluids interactions, among others. Also, in relation to temperature and pressure, there is uncertainty in the temperature of the well, in the dynamic properties of the pump, in the density and rheology of the drilling fluid. All this results in hook load sensors not always showing the correct hook load on drilling rigs.

Hoisting System - WOB Measurement

The main reason behind the attempts to get better estimates of drill string weight is that the WOB is determined indirectly from the measured weight of the drill string. The drilling process can be controlled in great part by the weight of the drill bit. It is still rare, and it is challenging to measure directly. Instead, we often estimate it indirectly.

The indirect WOB measurement is often estimated in the industry as a difference between a reference hook load (representing the off-bottom weight of the string) and the actual hook load, as well as a final physical correction of WOB. The reference hook load, also called tare string weight, is normally reset by the driller when the string is rotating off-bottom. It is assumed to be constant until the string is elongated by a new drill pipe and the bit is off-bottom for a new weight reading. This resetting of the reference hook load is often called the WOB zeroing [11]. This method assumes that the reference string weight is constant and does not change during drilling.

$$\text{WOB} = HKLD_{FREE-ROTATING} - HKLD_{\text{measured drilling}} \pm \text{Correction}_{\text{physics}}$$

Today's practice with a constant reference hook load suffers from several weaknesses, such as:

- The WOB zeroing procedure is manual and varies from driller to driller and from one stand to another. There is also a risk that the zeroing procedure is forgotten or done while the bit accidently is on bottom.
- It does not account for the changes in axial friction along the string while drilling.
- Changes in hydraulic buoyancy due to change of mud densities and cuttings in suspension are neglected.
- Changes in pumping rates and flow induced lift forces are neglected.
- Changes of surface buoyancy are neglected.

2.1.2 Rotating System

In today's drilling rigs, the industry standard for this system is Top Drive, which has replaced the old Rotating Table technology. This technology has major advantages such as fewer connections (drilling complete 3 joints stands), back-reaming, and an easier process of connecting to the drill string and resuming circulation [12].

Rotating Systems are capable of measuring two key measurements for modeling, namely RPM and torque. In the case of torque, it is defined by Johancsik et al. [13] as *"the moment required to rotate the pipe"*. The surface torque estimate is based on a friction measurement from the bit to the top drive.

2.1.3 Circulating System

In a mud circulating system, mud pumps, flow lines, standpipes, mud hoses, drill strings, bit nozzles, mud pits, surface mud processing, and other preparation equipment are the primary components.

It gives us two measurements that we commonly use for modeling: the flow rate and the standpipe pressure. This first measurement is indirectly determined by the number of strokes of the mud pump, efficiency, and piston diameter, or directly by flowmeters located before the standpipe. The second measurement is the total pressure loss inside the surface installations, drill string (Drill Pipe, BHA, Bit), and well annulus.

2.2 Data Collection

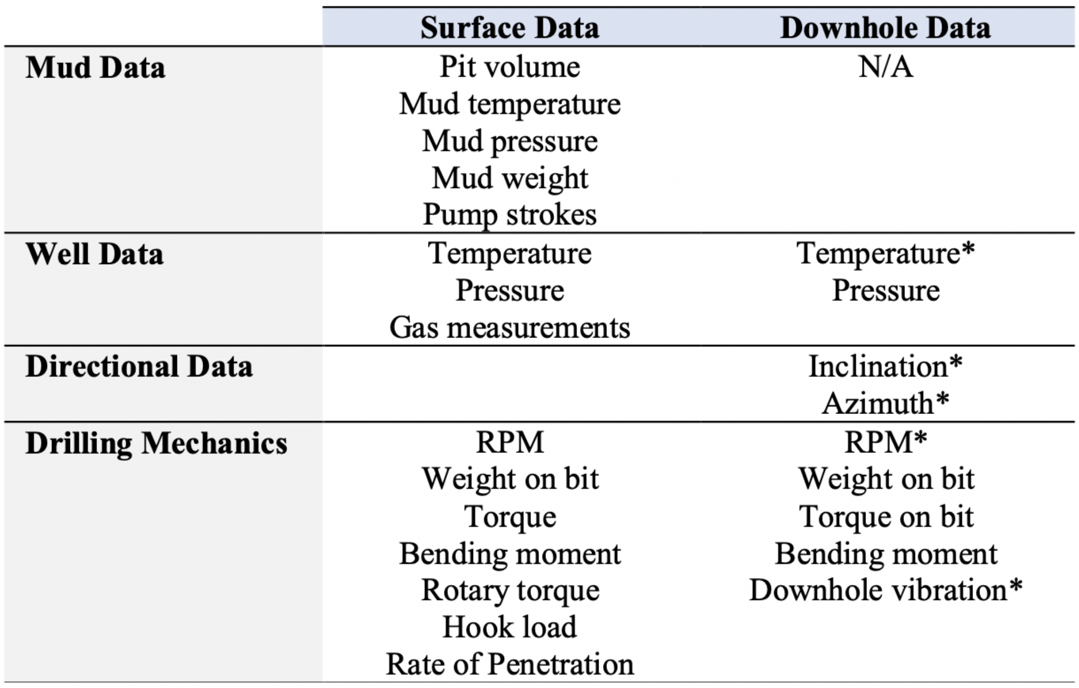
Over the last 50 years, data collection methods for both surface and downhole measurements have greatly improved. Oil and Gas drilling operations can now collect surface measurements of key drilling data, rotary speed, block position (for rate of penetration), hook load (for surface weight on the bit), mud pressure, and pump strokes (for flow rates). They can also collect downhole measurements of azimuth, inclination, temperature, pressure, revolutions per minute, torque on bit, weight on bit, vibration, and bending moments using an MWD device (although not necessarily in real-time) [14].

A thorough review of the surface and downhole data collection equipment used in the petroleum industry and its costs, capabilities, limitations, and impacts is of paramount importance. The next figure 2.3 will provide us with a brief summary.

2.2.1 Downhole Data Transfer

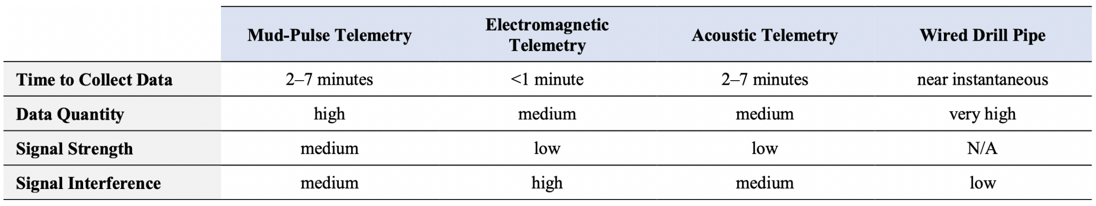
Data from downhole devices can be transmitted to the surface in near real-time or be stored on an external memory device for future review on the surface as depicted in figure 2.4.

Mud-pulse telemetry, the most commonly used tool, sends information to the surface. A small pressure wave is created within the mud, which is decoded at the surface and allows useful real-time information to be obtained. Positive or negative pressure pulses, as well as frequency-modulated waves, are generated using binary code. Also, to collect data in real-time, acoustic telemetry uses sound waves and amplifiers along the drill string. The drawback of this method is that it can't provide as much data as other forms of telemetry [15].



	Surface Data	Downhole Data
Mud Data	Pit volume Mud temperature Mud pressure Mud weight Pump strokes	N/A
Well Data	Temperature Pressure Gas measurements	Temperature* Pressure
Directional Data		Inclination* Azimuth*
Drilling Mechanics	RPM Weight on bit Torque Bending moment Rotary torque Hook load Rate of Penetration	RPM* Weight on bit Torque on bit Bending moment Downhole vibration*

Figure 2.3: Collection of data from surface and downhole tools



	Mud-Pulse Telemetry	Electromagnetic Telemetry	Acoustic Telemetry	Wired Drill Pipe
Time to Collect Data	2–7 minutes	<1 minute	2–7 minutes	near instantaneous
Data Quantity	high	medium	medium	very high
Signal Strength	medium	low	low	N/A
Signal Interference	medium	high	medium	low

Figure 2.4: Data Transfer

In addition, thanks to the development of electromagnetic telemetry (EMT), information is sent to the surface more quickly, and in less than a minute, the package of data is received. The method is much faster than mud-pulse telemetry but is not suitable for areas with excessively conductive or resistive soil.

In other cases, a wired drill pipe is used instead of telemetry. With this method, a wire runs along the inside diameter of a drill pipe to transmit downhole data directly to the surface in a near-instantaneous fashion.

2.2.2 Downhole Data Collection

There is no denying the importance of downhole measurement technology since theoretically, it offers a highly accurate source of information that can be trusted to make important decisions at the well site. Despite their benefits, however, mud pulse-based MWD systems have posed a barrier to the development of real-time monitoring of downhole drilling applications.

Wit the developments in technology, data transmission speed has dramatically increased since the introduction of wired drill pipe (WDP), an enhanced data acquisition technology. "WDP enables to transmit data 10,000 times faster than fast mud telemetry," stated Lesso et al [16]. In real-time data analysis, this enabled many much-needed improvements such as petrophysical properties, drill string positioning and directional drilling control, as well as drilling mechanics and dynamics.

Thus, collar-based enhanced measurement systems (EMS) based on wired drill pipe technology have been widely implemented for high-speed telemetry streaming. This tool is also capable of an array of measurements, including multi-axis vibration, load, pressure, bending, and rotation.

The *DWOB* and *DHTQE* values for the wells in our study are measured from the downhole measurement tool called enhanced measurement system (EMS), which is placed on top of the BHA. In recent years, the introduction of Wired Drill Pipe (WDP) measurement system technology has shown in some cases, especially in high-angle wells, a considerable difference exists between the measured *SWOB* and *DWOB*.

Unfortunately, the measurements from the *DWOB* sensor are inaccurate since non-readables are quite frequent, they are not tared and not fully temperature corrected [17]. In simple words, it can only show relative changes, and as a consequence, it's not a good alternative for the driller to guide himself.

A majority of downhole data is noisy and has some poor quality in some cases, as in our dataset. Transmissions from the downhole to the surface are inherently contaminated with various kinds of noise, which may be of much higher frequency or amplitude than that of the encoded signal. The received signal should be processed to identify useful components. In particular, the pump noise is the major source of noise during signal transmission.

2.3 Data Driven Modeling - Unsupervised Classification

A number of successful data-driven modeling applications have attracted attention from the Oil & Gas Industry, which is seen as the future in the long run due to the potential it presents to optimize drilling operations [18].

Today, Machine Learning is a powerful tool for developing data-driven or hybrid models based on large amounts of field data. The unwritten rule of ML is that simple models are more cost-effective and scalable than complex ones, most notably when we've got powerful and easy-to-use ML algorithms at our fingertips [19].

Usually, the domain experts must rely on manual labeling in cases where labels are not available from the system (e.g., reviewing recorded data and categorizing observed behaviors into relevant behavioral classes). Unsupervised Learning is used as an approach to the problem of laborious and error-prone manual classifications.

The purpose of an unsupervised classification algorithm is to detect structure in unlabelled data by discovering tendencies and similarities between the features selected from the preprocessed dataset. Several unsupervised ML algorithms were examined and tested to obtain clusters able to identify the different patterns and behaviors in the HKLD.

Clustering data based on relationships among the variables in the data allows a set of data to be categorized into classes. We aim to group data into clusters (regions) that are as similar as possible while still being as different as possible from one another. The classes are often called targets, labels, or categories. Our major goal here is to categorize the following: *"Detection of the period between the time we pick up the string until we tag the bottom"* to be able to differentiate between running in hole and start of drilling operations. The ability of unsupervised learning to handle large volumes of data in real-time is another important feature to highlight for its application in the industry.

2.3.1 Fuzzy c-means

Fuzzy c-means is a type of unsupervised clustering algorithm that permits us to build a fuzzy partition from data through soft clustering. On a scale of 0 to 1, every data point is found simultaneously in all clusters with a varying degree of membership. Conventionally, we classify each data point into the cluster it has the highest membership in. Data points with low membership have very little effect on the centroid position.

Fuzzy clustering is more robust against noise than other clustering techniques. Unlike a traditional clustering algorithm, fuzzy clustering is more robust against outliers and noise because low-scoring data points have a smaller impact on the cluster center position. A key aspect of soft clustering is the membership score, which can be used to filter genes that don't belong to any cluster.

Fuzzy c-means (FCM) was developed by Dunn [20] in 1973 and enhanced by Bezdek [21] in 1981, and it is commonly used in pattern recognition. In Fuzzy c-means, the objective function is minimized as follows:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^m \left\| x_i - c_j \right\|^2 \quad , \quad 1 \leq m < \infty$$

Where:

- x_i = is the ith of d-dimensional measured data.

- $u_i j$ = degree of membership of x_i in the cluster j.
- c_j = is the d-dimension center of the cluster.
- m = real number larger than one.
- $|| * ||$ = it expresses the resemblance between any measured data and the center.

Through an iterative optimization of the objective function shown above, fuzzy partitioning is performed, with every new membership $u_i j$ and new cluster centers c_j by:

$$\mathrm{u}_{ij} = \frac{1}{\sum_{k=1}^{C}\left(\frac{\left\|x_i - c_j\right\|}{\left\|x_i - c_k\right\|}\right)^{\frac{2}{m-1}}} \tag{2.1}$$

$$\mathrm{c}_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} \cdot x_i}{\sum_{i=1}^{N} u_{ij}^{m}} \tag{2.2}$$

We stop iterating when $\max_{ij}\left\{\left|u_{ij}^{(k+1)} - u_{ij}^{(k)}\right|\right\} < \varepsilon_i$, where k are the iteration steps and ε represents the termination criterion between 0 and 1. As a result of this procedure, we converge at a local minimum or saddle point of J_m.

The algorithm consists of the following steps:

- Initialization of $U = [u_{ij}]$ matrix, $U^{(0)}$
- Calculation of the centers $C^{(k)} = [c_j]$ with $U^{(k)}$ for each k.
- Update $U^{(k)}, U^{(k+1)}$.
- Stop iterating if $||U^{(k+1)} - U^{(k)}|| < \epsilon$; otherwise get back to step two.

2.3.2 K-Means clustering

K-Means Clustering is considered an unsupervised centroid-based ML algorithm [22]. The K-Means algorithm, unlike traditional supervised Machine Learning algorithms, attempts to classify data without first having been trained with labeled data. The algorithm consists of iterating over the dataset to divide the data into pre-defined distinct non-overlapping subgroups (clusters).

Each point in the dataset belongs to a single cluster. Data points are assigned to clusters in which the sum of the squared distance between them and the cluster's centroid (the average of all data points belonging to that cluster) is at its minimum. That way, the intra-cluster data points are as similar as possible while also making sure the clusters are as distinct as possible.

K-Means Clustering minimizes an objective function, in this case, a squared error function. In particular, the objective function is:

$$\text{J}=\sum_{j=1}^{k}\sum_{i=1}^{n}\left\|x_i^{(j)}-c_j\right\|^2 \tag{2.3}$$

Where:

- $x_i^{(j)}$: corresponds to each data point.
- c_j: indicates the distance of the data points from their cluster centres.
-
$$\left\|x_i^{(j)}-c_j\right\|^2$$
: measures the distance between the cluster centre c_j and a certain data point $x_i^{(j)}$.

The algorithm may be summarized in the following steps:

- Place K points on the clustering objects space where each point represents the initial group center.
- Choose the group with the closest centroid for each object.
- Recalculate the positions of K centroids after all objects have been assigned.
- Repeat steps two and three until the centroids remain stationary. By doing this, the objects can be separated into groups from which the metric to be minimized can be calculated.

2.3.3 Gaussian Mixture Models (GMM)

Clusters of irregular shapes are often observed in drilling data, and GMM supports them. Different clusters can be accurately grouped by GMM when distinct features are mixed together. Additionally, it accounts for variance, resulting in a probabilistic soft assignment of data points to clusters instead of K-means, which assigns data points to clusters based on a circular distribution around centroids. It is a distribution-based system rather than a distance-based system. Also, the data point is classified as a probability of belonging to one cluster or another.

In essence, the mixture distributions [23] generate properties that describe the overall data set as well as make inferences about the properties of the clusters present in the data space. Following that, the model learns the specifications of the different subclusters automatically and uses its learning to cluster the data points.

In GMM, the probability distribution function is defined as follows:

$$N(\mu, \Sigma) = \frac{1}{(2\pi)^2 \sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

Where:

- μ = average.
- Σ = Matrix of Gaussian covariance.
- d = Feature counts in our dataset.
- x = total number of datapoints.

A probability value X can be calculated for a given data point by multiplying the probability distribution function of the d-dimension by the previous probability of the Gaussian 2.5.

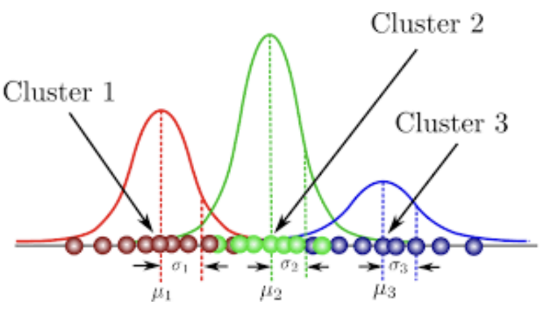


Figure 2.5: GMM Clustering Representation

2.3.4 Number of Optimal Clusters

Quality clustering occurs when data points within a cluster are apart from those in other clusters. In addition, one of the challenges we face when clustering is figuring out the optimal number of clusters to use in our algorithm.

The number of clusters (K) is chosen manually according to human insight or through a mathematical approach. If domain knowledge is not available, mathematical methods such as the Elbow method can be used to identify the right number of clusters. In general, a higher K value is indicative of smaller groupings and more granularity, whereas a lower K value is indicative of larger groupings and less granularity. A range of values of 'k' is fitted into the model using the Elbow method in order to determine the optimum value of 'k' (number of clusters). More specifically, the best number of clusters for a given problem is the one that provides the lowest inertia value with the minimum number of clusters possible.

As shown in the Elbow Method graph below, where the Within-Cluster-Sum of Squared Errors (WSS) is plotted in x axis and different values of k are plotted in y axis 2.7, the plot appears like an arm with a clear elbow when k=3. Sadly, data are not always so clearly clustered. Therefore, the elbow may not always be clear and sharp.

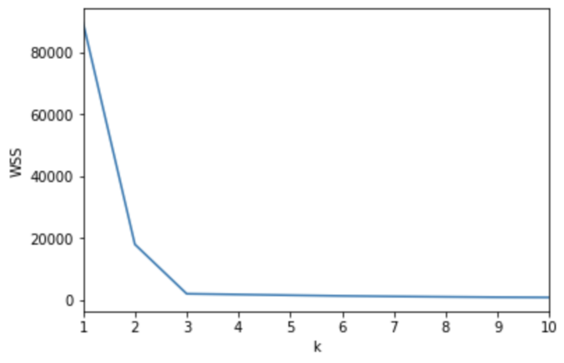


Figure 2.6: Elbow Method Representation

2.3.5 Clustering Evaluation

So far, different unsupervised ML approaches have been discussed. However, how should clustering algorithms be evaluated? Typically, unsupervised learning techniques are used when the clustering validation process is not available. In fact, clustering algorithms are not as simple to evaluate as supervised learning algorithms because their performance cannot be measured by counting errors or precision and recall. Consequently, external validation methods are not available for our ML approach.
It is important to keep in mind that the most common method of evaluating clustering algorithms is from a research and engineering perspective of the final result.

Internal validation Methods

In contrast, internal validation methods allow you to verify the quality of the clustering structure without having access to external information (i.e. they are based on input data). Internal validation methods typically combine cohesion and separation to estimate the validation score as represented in figure 2.7.

Validation scores are calculated for each cluster and then combined into a final score for each cluster in a weighted manner. Given C as a set of clusters, the validity of C will be calculated as follows:

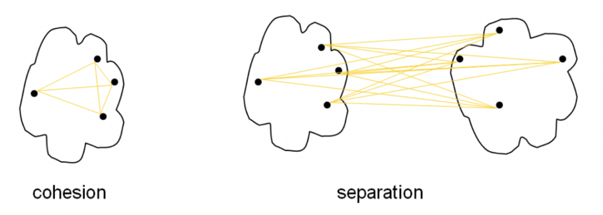


Figure 2.7: Internal validation

$$\text{Validity(S)} = \sum_{k=1}^{n} w_k * \text{Validity}\left(C_k\right)$$

As for the cohesion calculation, it is possible to compute a cluster's cohesiveness by summing the similarity between each pair of records.

$$\text{Cohesion}\left(C_k\right) = \textstyle\sum_{x \in C_k ; y \in C_k} \text{similarity(x, y)} \tag{2.4}$$

Furthermore, by summing up the distance between each pair of records belonging to two different clusters, it is possible to compute the separation between the two clusters.

$$\text{Separation}\left(C_j, C_k\right) = \textstyle\sum_{x \in C_j ; y \in C_k} \text{distance(x, y)} \tag{2.5}$$

Similarity or dissimilarity between clusters can be determined by measuring the distance between cluster points. If the clustering algorithm can separate dissimilar observations from each other and bring together similar observations, then it has worked well. Nevertheless, in reality, rather than dealing with two metrics, the most popular evaluation metric for clustering algorithms is a measure that combines them into a single one called Silhouette Score.

In the Silhouette Score, a data point is ranked based on how similar it is to other data points in its cluster relative to data points outside the cluster (the score is tallied over all data points). It examines how distinct clusters are in space. This method also has the advantage of not requiring prior knowledge of ground truth. It has a range from -1 up to 1. Closer to -1 suggests incorrect clustering, while closer to +1 indicates dense clustering.

$s = \frac{b-a}{\max(a,b)}$

Where:

- a: average distance between a sample and all the other points in a cluster.
- b: Known as the average distance between a sample and all the other points in the nearest cluster.

Chapter 3

Challenges

The floating rigs under study present additional challenges in determining the relative position of the hook (top of string) with respect to the seabed, as well as axial forces that act on the string. Such platforms as semi-submersible rigs are not resting on the seafloor. A semi-submersible rig is a floating platform that is moored in a systematic manner to withstand rough waters.

An analysis of the dynamics of our system is then quite important for understanding how the signals behave overall. The next step is to present the main factors involved for the rigs under analysis.

3.1 Rig-heave induced pressure variations downhole

Disturbations such as drillstring movements can pose a challenge to pressure control. Drillstring movements when running into or pulling out of the hole have long been known to induce pressure oscillations that can potentially violate pressure margins, especially for floating drilling units, where heave is disturbing the signal.

When drilling offshore from a floating rig, a form of drillstring movement occurs due to the wave-induced heaving motion of the rig. Signal analysis on semi-submersibles needs to consider the fact that surface measurements are influenced by both lateral (tidal movement) and axial (up-and-down movements of the rig, also known as rig heave) effects. We need to observe how the block moves in relationship with the hook load signal. One might be able to see through the heave disturbances, which appear periodically, and spot the signal characteristics necessary to diagnose the problem.

If the drillstring is extended by a segment, the heave compensators that normally decouple the drillstring from the rig's heaving motion are disabled, and the drillstring continuously oscillates with the waves [24]. For that reason, it is also important that the rig heave motions, and the heave compensation system are accounted for and its effect eliminated.

Firstly, the ADC system is given responsibility for computing the motion of the traveling block relative to the seabed. In that case, it must be capable of receiving signals for rig heave and the motion of the heave compensation system [25]. The ADC system must also receive information about the rig's position relative to the seabed. It's important to note that the hole depth is calculated by the ADC system as part of the bit depth calculations. Next, it is possible to estimate the rig motion with a motion reference unit (MRU) or by measuring the slip joint motions. With a Drawworks-based hoisting system equipped with a crown block compensator, position information obtained from the drawworks or wireline is normally used for measuring the string position relative to the crown block, whereas the crown block position is measured relative to the floating rig.

The system should therefore transfer the following information:

- The string length.
- The position of the traveling block or hook in relation to the crown block.
- The position of the crown block in relation to the rig.

To be able to estimate WOB from hook load measurements, it is also necessary to know the lift force from the heave compensator cylinders (or accumulator pressure). In addition, the moving environment complicates other types of measurements. For instance, liquid levels in tanks and return flow rate measurements.

When a Crown Block (CB) is compensated in a Draw-works hoisting system, it is also important to know where the compensated CB is relative to a reference position (for example, the water table where the CB is landed when compensation is turned off). Additionally, we propose that the lifting force of the compensator system be provided so that an ADC system can discern whether CB-motions are due to altered surface lifting capability or to downhole conditions [25].

3.2 Friction force contributions from string motions

Friction factor uncertainty includes tortuosity, mud properties, fluid viscous effect, cutting bed, formation instability, borehole temperature and pressure, non-uniform geometrical interactions between borehole and drillstring, etc [26]. Especially in horizontal wells with long-reach, excessive axial drag can be problematic. If the drillstring is not rotated during tripping operations, the drag forces will be higher. In contrast, drill string rotation reduces friction and drag, which is extremely important for operating safely and efficiently.

3.3 Inclination of the wellbore

The larger the inclination, the greater the influence of changes in the downhole situation on the hook load since the force of the drill string against the wellbore wall increases. Furthermore, overpulls may also appear random and abrupt due to heave, while the peak value may not accurately reflect the difference in hook height [27].

3.4 Cuttings Bed Build Up

To prevent cuttings from beding inside the hole, drill cuttings should be circulated out of the wellbore. If the cuttings are not removed, the drillstring can pack off - another type of stuck pipe. Accumulation occurs when the drilling fluid does not circulate fast enough, or it is not viscous enough to overcome the drag forces on the solids. Rather than flowing up and out of the hole, solids flow downwards 3.1.

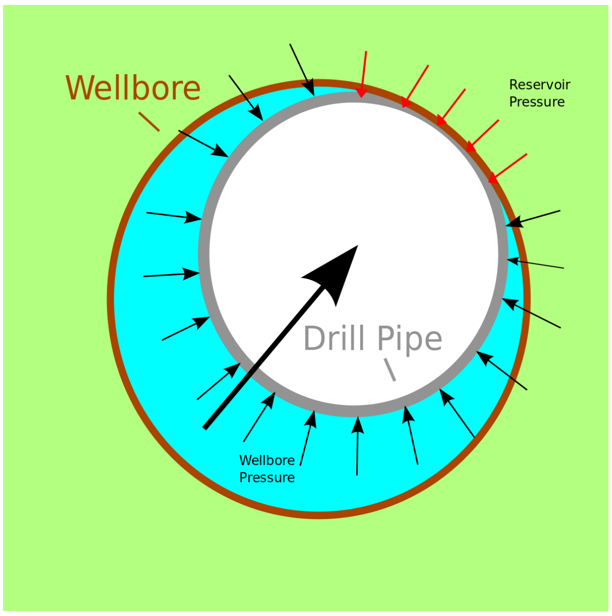


Figure 3.1: Differential Sticking

The buildup of cuttings in the annulus is a major problem when drilling deviated or horizontally. Cuttings on the drill string result in an increase in friction that persists until the drill string becomes stuck and the hole cannot be drilled. This cutting´s build-up can be detected very early, if automated, real-time monitoring system continually controls the trend of the measured hook loads for the stand currently being drilled off. As a result, the drilling team can decide on the right countermeasure at a time when there is still time to avoid costly trouble[28].

3.5 Differential Sticking

It is very common in the drilling industry to experience differential-pressure sticking. It occurs when the drillstring becomes stuck to the filter cake due to the pressure difference between formation and wellbore during a differential sticking, as represented in figure 3.1. Drill string friction with wellbore walls is increased by differential pipe sticking [29].

This typically happens in depleted zones or permeable zone characterized by a high loss of circulation. In the case scenario, the hydrostatic pressure of the mud is much higher than the formation pressure, the drill string is forced into the filter cake. When the drilling fluid losses to the formation, it leaves behind the solid-phase [30]. As a consequence, this solid-phase settles onto the side of the borehole wall, and the formation of filter cakes made from these solids can become very thick and sticky. The normal force of the drill string against the wellbore increases when the wellbore pressure differs from the formation pressure. This can result in a stuck pipe situation. Especially since filter cake has a high friction coefficient, which causes the force needed to move the drill string to increase, sometimes to levels above its strength limit.

The differential pressure can be expressed as $\Delta\rho = \rho_m - \rho_{ff}$

The monitoring of out-of-slips drillstring weights can provide early warnings about stuck pipeline situations. For each stand drilled, out-of-slip weights are plotted in a hook load over a measured depth plot on the same depth as the median hook load values. In this way, the user can identify the trend of increasing out-of-slips weights very quickly. As a result, the user can respond appropriately and avoid expensive sidetracks. Also, a sudden increase in the peak HKLD marks when differential sticking begins, compared to normal operations where HKLD while moving the string only varies very slightly [31].

3.6 Drillstring weight

Once one stand is drilled, it is a good practice to allow the cuttings generated to circulate above the BHA, depending on the operator policies this step might be skipped, but regardless of this, to add a new stand to drill, the top drive is lowered, pumps shut down, and the top drive disconnects from the drill pipe. After this, the *"hook load"* registered by the sensor is only the weight of the traveling block.

In contrast, when the drill string weight while running in the hole is more challenging to measure. As a result of frictional forces, this weight is less than the free-rotating HKLD [32]. The HKLD has a lower value than the actual weight of the drill string when measured from the weight indicator at the surface while the drill string is tripped in the hole.

When a drilling activity is started, the bit is typically a few meters above the bottom. The pump system, as well as constant rotation, are either started simultaneously or at different times, varying for every stand. When circulation is activated, the drill string experiences a buoyancy effect, which decreases the hook load. The presence of mechanical equipment like float subs plays an important role when buoyancy builds up. Fluids cannot enter or recede into the drill string with this kind of equipment. If the drill string has to be filled with mud from the top, this is especially crucial when tripping into the hole. As the fluid exits the hole, it may not be able to flow back in, filling the drill string completely.

Finally, when we tag the bottom, the string begins to compress, and the HKLD is gradually transferred to the bit. Consequently, we must account for the force transmission delay downhole that is caused by compression.

Chapter 4

Data analysis

4.1 NCS Data sets

This study uses confidential and proprietary data owned by SEKAL AS, a leading provider of drilling software technology company that strives for powering autonomous and safe drilling operations.

The data sets provided contains real-time drilling data both in time-based and depth-based data from several semi-submersible North Continental Shelf wells. The information was downloaded in a comma-separated value (.csv) format archive so that the handling and selection of the most appropriate well to execute this study was easier. Additional data about the wells were also taken into account for evaluation, including the summary of drilling activities, operations, incidents, and general remarks from daily drilling reports.

4.2 Exploratory Data Analysis (EDA)

From problem definition to result interpretation, domain knowledge plays an important role in data analysis. In turn, it will give us an insight into how is data are collected and, therefore, how to preprocess them. Therefore, understanding our data in the context of the problem we're trying to solve before we move forward with modeling is essential. As there are many drilling parameters, if they are all included as independent inputs, modeling would be both computationally expensive and inaccurate.

Therefore, at the start, we will perform an exploratory analysis to evaluate the raw data quality and quantity for several selected wells. Coding work was done using Python [4] and the Pandas library [33].

4.2.1 Importing and Visualizing the Data

Importing and visualizing data is the first step in exploratory analysis. For the well we are demonstrating the results in this Master Thesis, there are 32 different features and 220.000 rows in the data set. It is observed that there exists redundancy as well as interdependency among various data collected. Thus, it is of utmost importance to select the most relevant features according to their influence on the problem's objective.

Then for feature selection, a histogram plot is required to analyze the data. An analysis of the data through a histogram plot is necessary. There are various techniques for this purpose; the first one that we will analyze is the histogram plot presented in figure 4.1. The histogram plot presented in figure 4.1, shows in the x axis the range of values for each feature and in the y axis the frequency of such values.

A total of twelve features are selected, and it is verified that the selected parameters are stored in the appropriate data type (integer, float, Python object, etc.). Below is a list of the features selected to be evaluated in this study 4.1 with the appropriate format and units for the measurements *'float64'* and for time *'datetime64[ns]'*.

The parameters are next described:

- *Time*: describes the exact moment at which the parameters investigated were measured (year, month, day, hour, minute and second).
- *Depth Bit*: corresponds to the bit position.

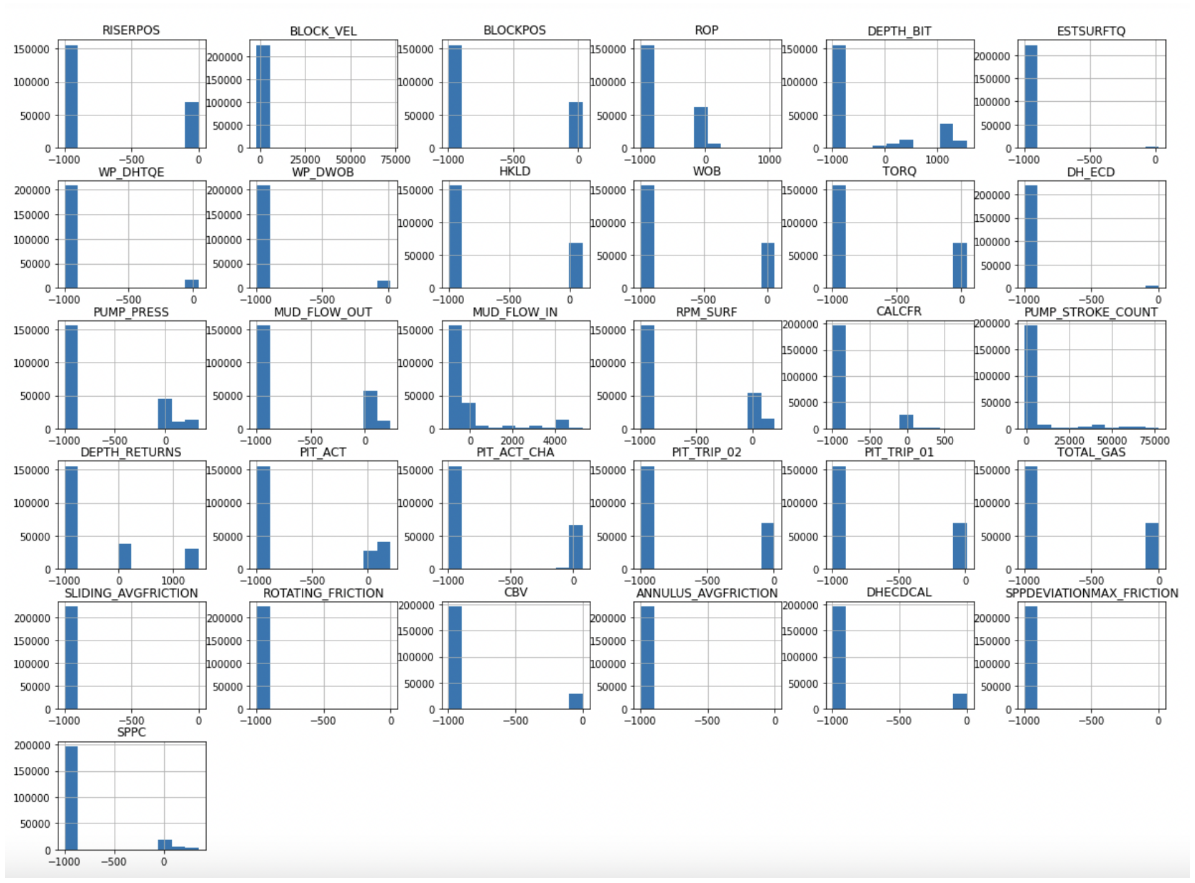


Figure 4.1: Histogram of features selected.

Number	Feature	Units	Measurement system	Data Type
1	Time	seconds	Direct	datetime64[ns]
2	Bit Depth	meters	Direct	float64
3	Average Hook load	tonnes	Direct	float64
4	Block Velocity	m/minute	Direct	float64
5	Block Position	meters	Direct	float64
6	Estimated Rate of Penetration	m/h	Calculated	float64
7	Weight on Bit	tonnes	Calculated	float64
8	Average Surface Torque	kN-m	Direct	float64
9	Average Rotary Speed	rpm	Calculated	float64
10	Pump Pressure	bar	Calculated	float64
11	Mud Flow In	L/min	Direct	float64
12	Equivalent Circulating Density	g/cm3	Calculation	float64

Table 4.1: Features selected for the different wells under study

- *Average Hook load*: identifies the actual weight of a pipe string as measured at the surface, affected by factors like buoyancy, friction, and others.

- *Velocity of block*: the speed at which we reel in (or out) drilling line from a spool to lift or drag a drill pipe.

- *Block Position* indicates the position of a traveling block.

- *Rate of Penetration* refers to the speed at which a drill bit penetrates the rock.

- *Weight on Bit*: downward pressure applied to the drill bit.

- *Average Surface Torque*: amount of rotation required to rotate both the drill string and the bit on the bottom of the hole.

- *Average Rotary Speed*: number of times the Top Drive makes one full revolution in one minute.

- *Pump Pressure*: represents the total pressure loss caused by the well surface equipment, the drill pipe, the drill collar, the drill bit, and annular friction losses around the drill collar and drill pipe.

- *Equivalent Circulating Density*: result of the hydrostatic pressure of the static fluid column in combination with friction pressure.

- *Mud Flow in*: volume of mud passing through the wellbore per unit of time.

From evaluating the plots presented in the raw data, the presence of outliers is evident and they need to be removed. As observed in figures 4.1 and 4.2, there is information from the parameters that need to be investigated, but it is imperative to ensure that the data is stored sequentially and no missing values are present.

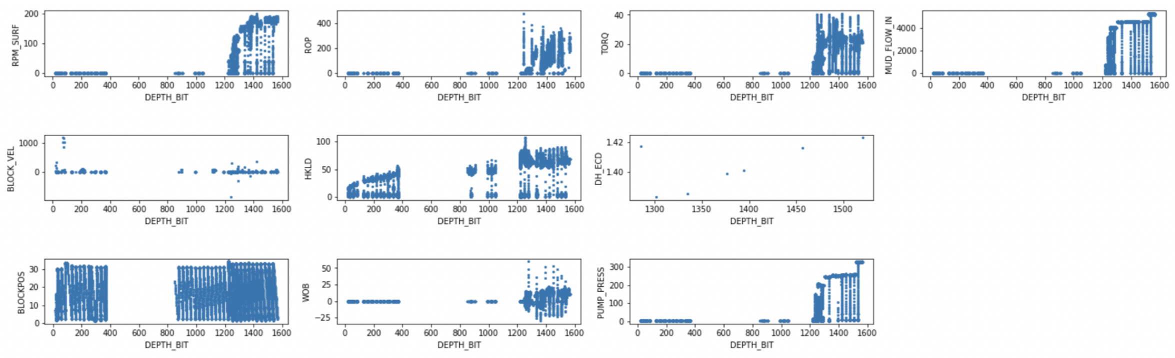


Figure 4.2: Raw dataset of the features selected.

We analyze the entire set of data from the well. Thereafter, it is required to identify appropriate sections of the well to work with.

4.2.2 Synchronization of timestamps

Each well has a time-based database, and each physical parameter (any feature other than time) has its own sensor and therefore its own time feature. The difficulty in working with different time columns makes it necessary to merge all the dataframes based on the time column with a higher number of measurements to prevent deleting relevant data.

In almost all columns of the data, there are many missing values. Data from different sources is automatically aligned. To align the recorded data, we represent them in our dataset with the value *'-999.25'*. The frequency of the measurements on the different features varied, so it was obvious that some variables had more measurements than others.

4.2.3 Data range selection

Before applying any data cleaning techniques, we can reduce the difficulty of data collection by applying proper domain insights. There is clearly a great deal of data inside this well section. This is because various drilling operations (e.g., drilling, tripping, casing running, etc.) are all recorded for the different depths [34].

When drilling, we pay particular attention to the connections. Now, we need to select a range of data that contains representative values for operations carried out during drilling operations, and then apply the corresponding data cleaning techniques.

From reading the daily drilling reports and analysing the data, it was discovered that of the different well sections, the [*1200- 1600*] meters section drilled on *23-12-2021* for *3 hours and 10 minutes* held the most relevant information for the study. Afterwards, an investigation of the data contained per depth was conducted through the Figure 4.3.

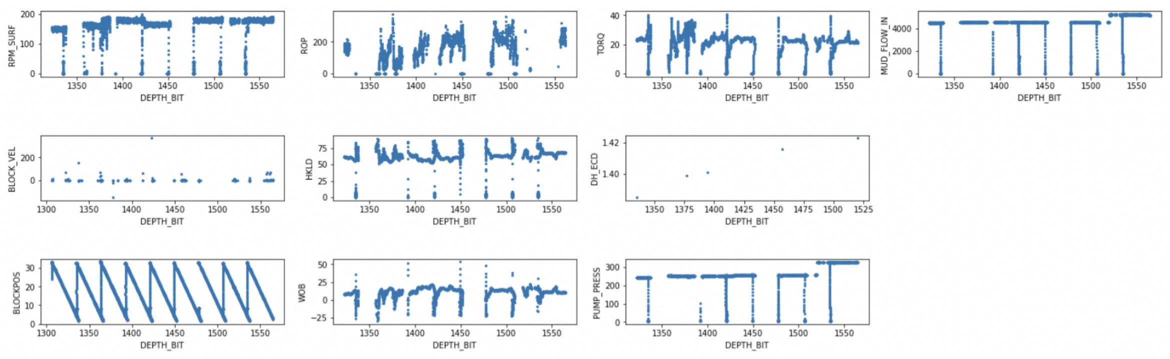


Figure 4.3: Raw data distribution of the 1200-1600 meters section

4.2.4 Evaluation of hook load trend after data set reduction

According to Table 4.1, the first feature selected for this investigation was *"Time"* because it enabled analysis of the data over time. As illustrated in the figure 4.4, the Hook load measurement data points are distributed based on the time, as well as depth. *Why is this important?*, it is important because this allowed to see the selected data that was generated during the drilling operations and make-up operations as we can see while the pipe is in-slips and out of slips. By doing this, we were able to remove most of the additional data points corresponding to other operations.

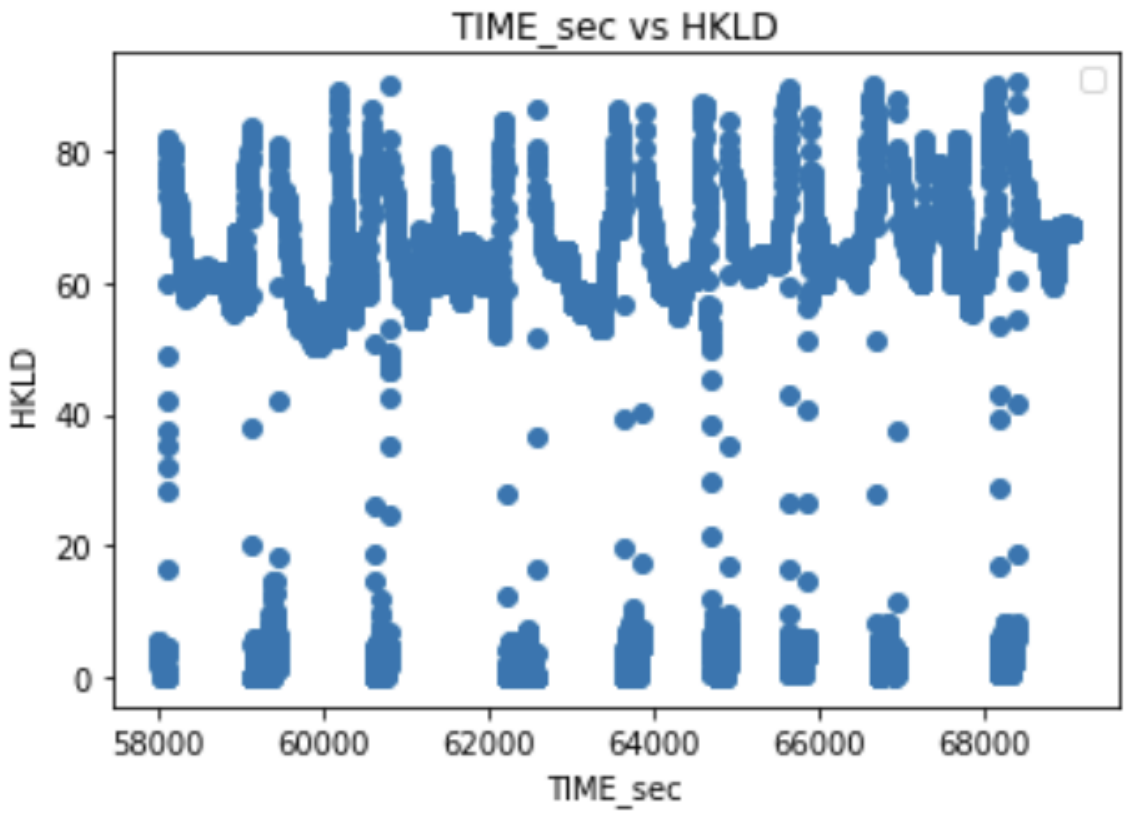


Figure 4.4: Raw data of the HKLD sensor of the 1200-1600 meters section

4.3 Data Cleaning

The quality of the data generated is directly proportional to its value in predicting drilling issues and uncovering hidden knowledge. As sensors, data storage, and computational power drop in price, we can collect more and more data. But if the data from sophisticated sensors is unreliable, the whole purpose of having additional sensors is defeated. Especially for drilling data, the methods for improving data quality play an ever larger role because of its poor data quality [35].

Preprocessing the data is the first step toward creating our first model. The preprocessing of recorded data, combined with the recognition of different patterns of operations, helps to reduce the amount of data that needs to be stored by condensing the amount of information for our analysis.

We will conduct comprehensive data cleaning and wrangling in preparation for next statistical approach, unsupervised learning and developing subsequent classification predictive models. Our model will learn from high-quality data if we follow this step. After all, *"a model is only as good as its data"* . In the following subsections, we will describe the steps involved in preprocessing the data and the principal problems faced while working with this data set will be presented.

4.3.1 Missing Values Handling

Our data sometimes has missing values in some columns due to mainly a sensor failing to read a certain value and high frequency of the sensor's measurement. When it comes to the "NaN" values, there are two ways to handle them: eliminate the entire row or fill in the missing information. Based on the size of the data set, one should decide what to do. In this study, removing the whole row was not an option, since it could potentially wipe out relevant information from other measurements that were available. Where there are missing values in the data, imputation is performed to retain as much data as possible replacing non-readable measurements marked as *'-999.25'* with *'NaN'*.

4.3.2 Resampling

Value acquisition for every feature is not fixed at a specific time stamp. The reading frequency in our dataset ranges from *+1 Hz* to *+10 Hz*. Resampling is performed to deal with this highly unbalanced datasets. Time series with constant time steps make finding patterns in the data easier. To find patterns in the data, most analytical methods use constant time steps. Thus, to obtain constant time-step sensor readings, implementing resampling is key.

As a first step, we removed or added samples from the majority *oversampling* class or minority *undersampling* class based on the range of time where they are located using a fixed period of 0.5 seconds *(+2 Hz)*. Then, we decided to aggregate them with a median function, since the median value is less sensitive to accidental peak measurements in the data stream.

4.3.3 Filling missing data

In most cases, gaps in the dataset are the cause of issues. Missing data must therefore be filled in. When estimating missing values, two different types of interpolation can be used: interpolation (linear, quadratic, and cubic) or filling the missing value with the last known value using forward or backward interpolation [36]. Each technique was evaluated.

Interpolating along a straight line is the simplest technique because two points are connected by a straight line, as defined in Equation 4.1 where b_0 is the intercept and b_1 is the slope of the line.

$$f_1(x) = b_0 + b_1(x - x_0) \tag{4.1}$$

Depending on the distribution of the data, quadratic or cubic interpolation can also be used. After evaluating the different options, linear interpolation was selected as the best option because of the high sampling rate from the sensor after resampling.

Afterward, linear interpolation is evaluated for different parameters. We create various gap sizes and thereafter, impute the missing data using linear interpolation. As gap sizes increase, both actual and evaluated metric scores for data imputation decrease, and then the R2 scores are calculated. There is also evidence that the performance of linear interpolation varies according to the feature. Due to this, each signal and feature is evaluated separately for various gap sizes.

4.3.4 Removing Outliers

This study removed outliers in order to narrow data to a realistic range. Visualization of the data indicates a need to delete some unnecessary information. In the beginning, data identification and observation of anomalies that are out of range can be done manually if they are easily recognizable. Consequently, we deleted drilling operations whose behavior exhibited a non-realistic behavior. The next step is to evaluate more advanced techniques to remove outliers: Interquartile Range (IQR) and Moving Average Filter.

Interquartile Range (IQR)

Outliers located far from the observation point can be removed using this method. There is one issue to consider when working with IQR (Equation 4.2) because it is aggressive when removing outliers. Hence, it should be handled carefully. Below is the mathematical procedure to remove outliers through IQR:

$$IQR = P_{75} - P_{25} \tag{4.2}$$

Where,

$$LowerRange = P_{25} - 1.5 * IQR \tag{4.3}$$

$$UpperRange = P_{75} + 1.5 * IQR \tag{4.4}$$

Therefore, a point outside of the range will be considered an outlier and removed. Outliers exist when data values fall outside the range defined by Equations 4.3 and 4.4, which are functions of P_{25} and P_{75} (that represents the 25th and 75th percentile of data points, respectively). We noticed that representative values from the dataset had been deleted following the IQR outlier removal method. For this reason, we disregarded the IQR method in our analysis.

Moving Average Filter

Moving average filters are low-pass filters that are commonly used to smooth data signals. It takes the average of a number of samples according to a specified filter window and produces a single output point. Obviously, this depends on the number of samples to be fed at once. As outlined by Sui [37], the model consists of:

$$y\left(t\right)=\frac{1}{n}\left(x_{(t-n+1)}+x_{(t-n+2)}+....+x_{(t)}\right) \tag{4.5}$$

A sample is defined as $x_{(t)}$ at a specific time t and as n samples. Using the Fourier transform:

$$x\left(t-k\right)\underset{\leftrightarrow}{}X\left(j\omega\right)e^{-jk\omega} \tag{4.6}$$

In frequency domain, the previous equation becomes:

$$Y\left(j\omega\right)=\frac{1}{n}X\left(j\omega\right)\left(e^{-j\omega(n-1)}+...+e^{-j\omega}+1\right) \tag{4.7}$$

In the end, the transfer function is:

$$H\left(j\omega\right)=\frac{1-e^{-j\omega n}}{n\left(1-e^{-j\omega}\right)} \tag{4.8}$$

With a higher value of n (window), more high-frequency noises will be blocked, but on the other hand this will also increase the time delay. As a result, it is necessary to make a compromise between noise removal quality and time delay. Then, we evaluated different values of n to see how delay and smoothing of the data affected the results.
The next figure 4.5 displays the preprocessed data after missing data have been filled in and outliers removed.

4.4 Feature Scaling - Normalization

To visualize the correlation between the different variables in our data, *feature scaling* needs to be conducted after preprocessing.

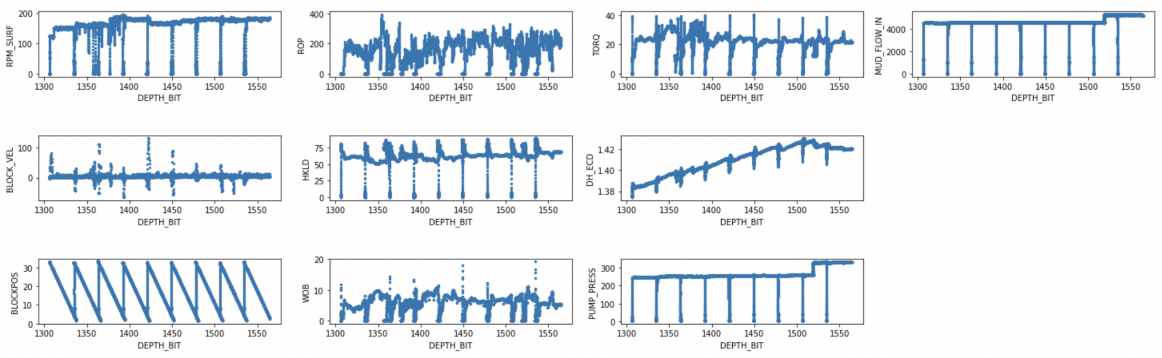


Figure 4.5: Preprocessed data

As a scaling method (often referred to as normalization), *min-max* scaling approach is employed. It scales the values from 0 to 1. Normalizing a data point means subtracting the minimum value in the data set, then dividing this value by the difference between its maximum value and its minimum value (Equation 4.9) [38]. The MinMaxScaler transformer from Scikit-Learn [39] was used for this [40].

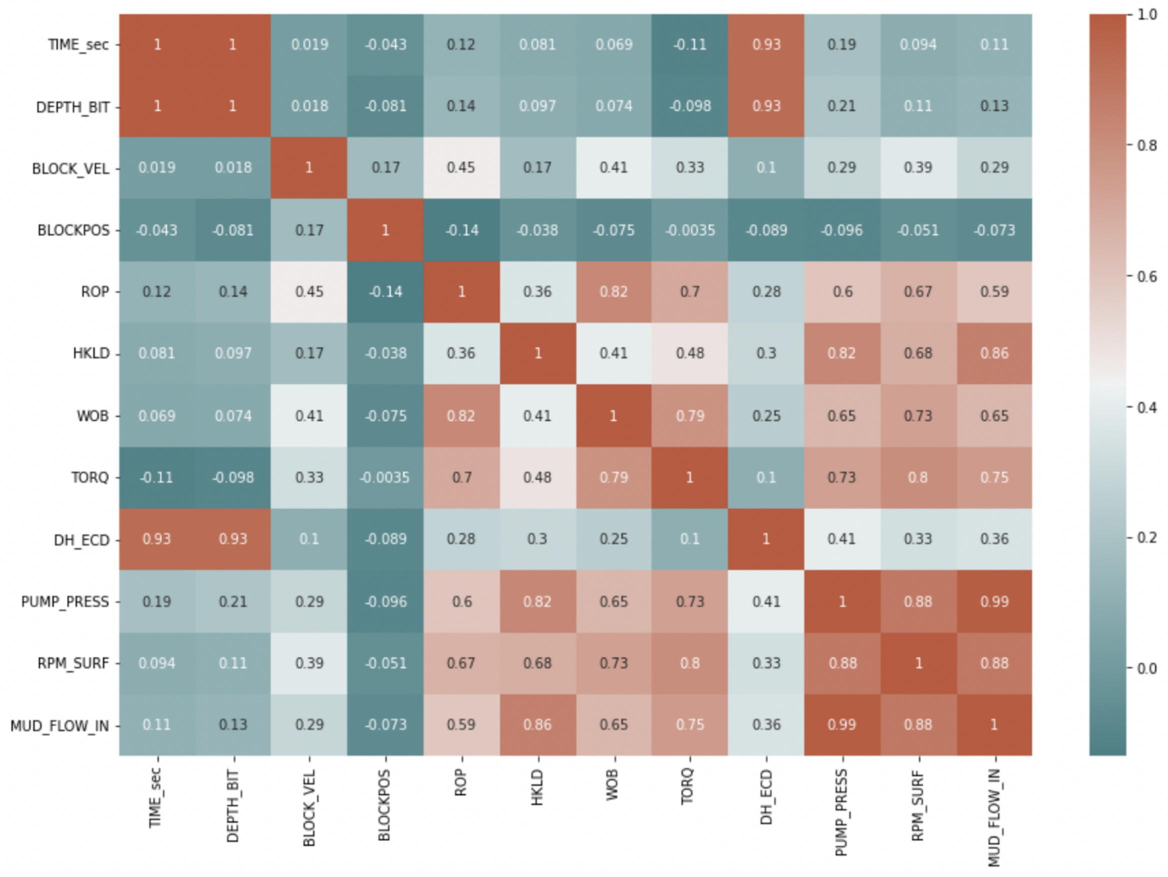
$$x^{(i)}_{norm} = \frac{x^{(i)} - x_{min}}{x_{max} - x_{min}} \tag{4.9}$$

As well, this normalized data approach will be used to feed scaled data to the unsupervised ML algorithms. The reason is that some Machine Learning algorithms don't perform correctly if the inputs have different scales, especially when dealing with classification algorithms. So, it should be standardized and clean.

To properly describe this issue in terms of drilling activities, we use different units and scales when measuring parameters. As an example, for one of our data sets HKLD measurements ranged from 0 to 100 Tonnes while bit depth ranged from 1200 to 1600 meters in the analyzed section and torque measurements ranged from 0 to 40 KN-m.

4.4.1 Heatmap

In addition, we used the heatmap to visualize data after both normalizing and processing the data. The next figure 4.6 presents the correlation indexes between the Hook load and the features selected for this study. From left to right, the diagonal group of squares have the darkest colors, as they represent intersecting features. This corresponds to a Pearson correlation coefficient of one, which is the maximum value. Depending on the degree of correlation between features, the intensity of the color will decay or not.



	TIME_sec	DEPTH_BIT	BLOCK_VEL	BLOCKPOS	ROP	HKLD	WOB	TORQ	DH_ECD	PUMP_PRESS	RPM_SURF	MUD_FLOW_IN
TIME_sec	1	1	0.019	-0.043	0.12	0.081	0.069	-0.11	0.93	0.19	0.094	0.11
DEPTH_BIT	1	1	0.018	-0.081	0.14	0.097	0.074	-0.098	0.93	0.21	0.11	0.13
BLOCK_VEL	0.019	0.018	1	0.17	0.45	0.17	0.41	0.33	0.1	0.29	0.39	0.29
BLOCKPOS	-0.043	-0.081	0.17	1	-0.14	-0.038	-0.075	-0.0035	-0.089	-0.096	-0.051	-0.073
ROP	0.12	0.14	0.45	-0.14	1	0.36	0.82	0.7	0.28	0.6	0.67	0.59
HKLD	0.081	0.097	0.17	-0.038	0.36	1	0.41	0.48	0.3	0.82	0.68	0.86
WOB	0.069	0.074	0.41	-0.075	0.82	0.41	1	0.79	0.25	0.65	0.73	0.65
TORQ	-0.11	-0.098	0.33	-0.0035	0.7	0.48	0.79	1	0.1	0.73	0.8	0.75
DH_ECD	0.93	0.93	0.1	-0.089	0.28	0.3	0.25	0.1	1	0.41	0.33	0.36
PUMP_PRESS	0.19	0.21	0.29	-0.096	0.6	0.82	0.65	0.73	0.41	1	0.88	0.99
RPM_SURF	0.094	0.11	0.39	-0.051	0.67	0.68	0.73	0.8	0.33	0.88	1	0.88
MUD_FLOW_IN	0.11	0.13	0.29	-0.073	0.59	0.86	0.65	0.75	0.36	0.99	0.88	1

Figure 4.6: Heatmap of the features under consideration

4.4.2 Probability Distribution

Drilling data is rarely distributed and intuitively understandable due to its skewed nature. In any case, data that follows certain distributions can be valuable, and identifying the probability distribution of its input parameters is critical for evaluating the most representative input parameters for our system to be fast and accurate [41].

For better understanding how some of the features considered in our analysis are distributed, Figure 4.13 shows a histogram with logarithmic scales.

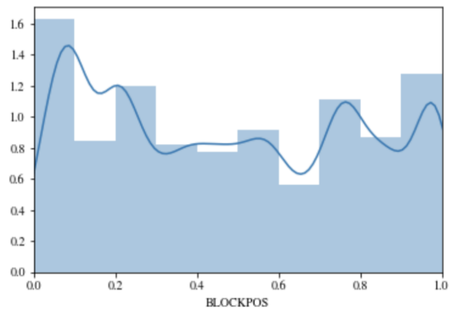


Figure 4.7: Probability distribution of the block position after data cleaning

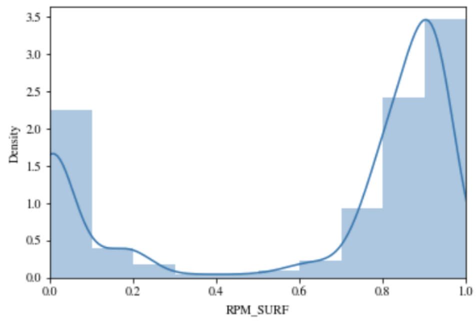


Figure 4.8: Probability distribution of RPM

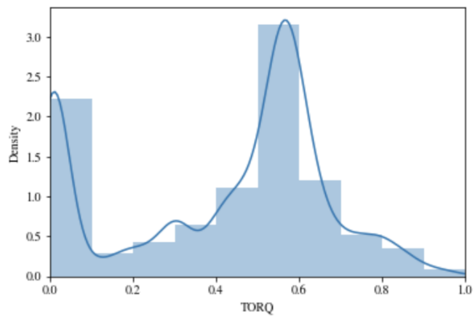


Figure 4.9: Probability distribution of Torque

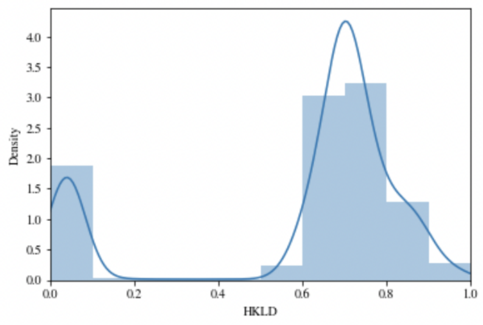


Figure 4.10: Probability distribution of HKLD

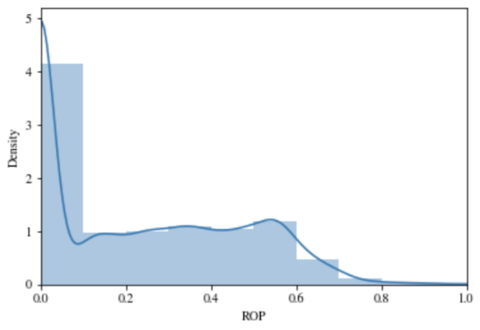


Figure 4.11: Probability distribution of ROP

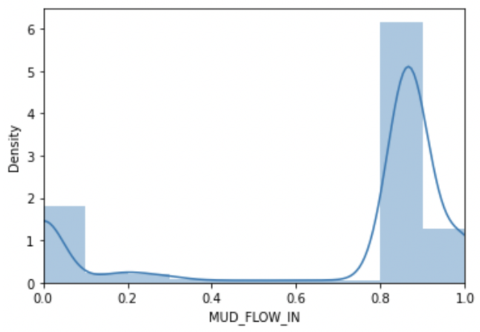


Figure 4.12: Probability distribution of Mud Flow

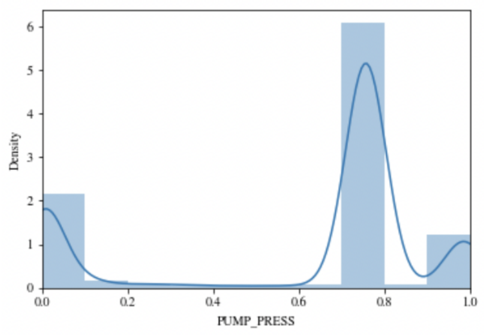


Figure 4.13: Probability distribution of the Pump Pressure

4.5 Exploratory Data Analysis (EDA) after data-cleaning

We can now begin exploring our data to find out more about it as soon as it has been cleaned. Following the data cleaning, I performed additional graphical EDA on top of quantitative EDA to look for trends and off behaviors on the preprocessed data. Now, we are able to understand what each sensor reading is telling us.

Chapter 5

HKLD Pattern Recognition

Slips are used to grip and suspend drill strings during drilling operations. In-slips operation occurs when the slip device is around the pipe during a connection. Upon driving down the stand, the driller stops rotation, and the slip device is placed around the pipe to take up the drill string's weight. With the tongs, the connection is broken, and the next stand is picked up from the monkey board. Once the slips devices pull up to go out-of-slips, the driller rotates the pipe and activates the pumps to proceed with the drilling.

When considering the sensor-based time-based data of the key variables selected for our study, the drillpipe status can be determined visually by taking into account the following factors:

- When the drillpipe is in-slip, the hook load goes from high to low, and on the contrary when it is released.
- Pump pressure or flow rate sensors can be used to detect whether the pump is on or off. When we go in-slip, rotation goes to zero.
- The rotation measurement is observed. When the drillstring goes in-slip, flow is stopped.

5.1 In-slips detection

Changes are easily spotted by the human eye, but are harder to identify using traditional statistical techniques. Instead, we use a pattern recognition statistical approach, to determine the status of the drillstring.

The algorithm produces two dimensionless classification values:

- *'0'* when the drill string is out-of-slips.
- *'1'* when the bit is in-slips.

Finally, to move on to the next phase of our research, we narrow down the dataset to only the *'Out-of-slips'* state after an automatic detection has been performed.

Throughout the next few subsections, we'll discuss both the methodology used as well as the findings of the pattern detection process.

5.1.1 Methodology

Based on the sensor's preprocessed data, the detection will infer the slip status. Here are two approaches investigated to detect this rig condition:

- Interquartile Range (IQR): traditional statistical approach.
- Change Point Detection (CPD): pattern recognition approach.

Following the application of these two different approaches to the grouping of data, the recognition of the status will be marked in different colors based on the dimensionless classification values mentioned before where:

- In-slips rig state is marked in red.
- Out-of-slips rig state is marked in blue.

5.1.2 Results - Interquartile Range (IQR)

An Interquartile Range (IQR) is calculated by dividing a data signal into four equal parts and measuring the statistical dispersion of the data in each part, or quartiles. Our goal is to find *'outliers'* in the data which represent our detected patterns of interest.

Below is a step-by-step procedure to follow:

- IQR measure of the difference between the 75th percentile (Q3) and the 25th percentile (Q1).
- Set upper and lower limits for the outlier.
- Flag any data points that do not fall within the upper and lower bounds as outliers.
- In the last step, plot the outliers over the time series data.

As illustrated in the following figure 5.1, the red-colored points represent the patterns within the upper and lower bounds of the hook load signal.

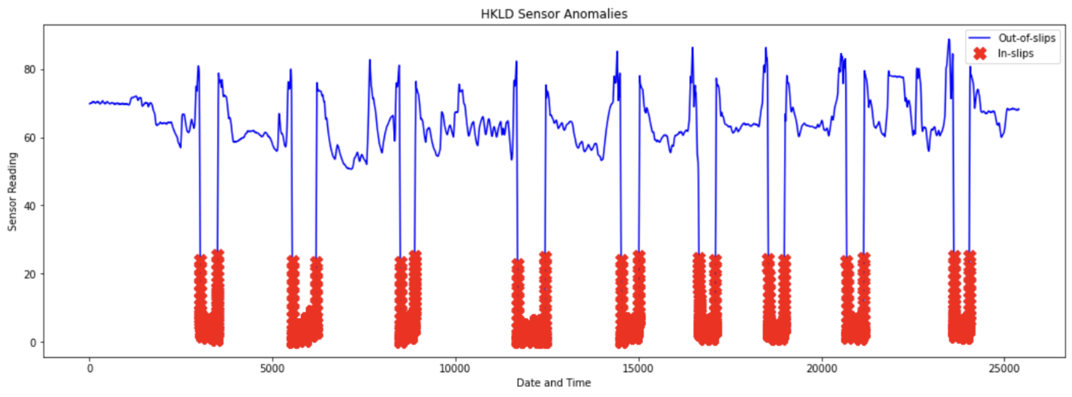


Figure 5.1: IQR - In slips detection

Above, it can be seen that the IQR method was not successful at detecting the patterns of interest and sudden changes in the trend for the entire range of data, every time when we went in-slips.

5.1.3 Results - Change Point Detection (CPD)

It is therefore necessary to develop a method capable of detecting abrupt or gradual changes in the distribution of data over time through several pattern recognition approaches.

In data time series forecasting, a change-point is an abrupt or structural change in the distributional properties of the data. In other words, it shows when the probability distribution of a stochastic process or time series changes. Change Point Detection (CPD) methods identify the changes in the trends and properties of time series data in the underlying behaviour of the system.

With CPD, fast signal segmentation can be performed and ruptures can be detected, which is perfect for detecting patterns of interest within the hook load time series signals. Actually, this problem is equivalent to the problem of segmenting time series.

Several CPD algorithms have been designed, enhanced, and adapted for change point detection. As the number of drops of the hook load value when we go into slips mode is known visually, we enter it as input in the chosen method *"Binary Segmentation Search Method"* to detect change points as it's more accurate than other CPD methods.

From figure 5.2, we may conclude that the operational status can be more accurately distinguished from each other using CPD compared to IQR method.

For other wells with more complex patterns, however, the preprocessed HKLD data structure makes it impossible to identify a trend change through *'Binary Segmentation Search Method'*. Rather, *"Dynamic Programming Search Method"* is implemented for change-point detection 5.3. The disadvantage of this accurate method is that it is much more computationally expensive than the former.

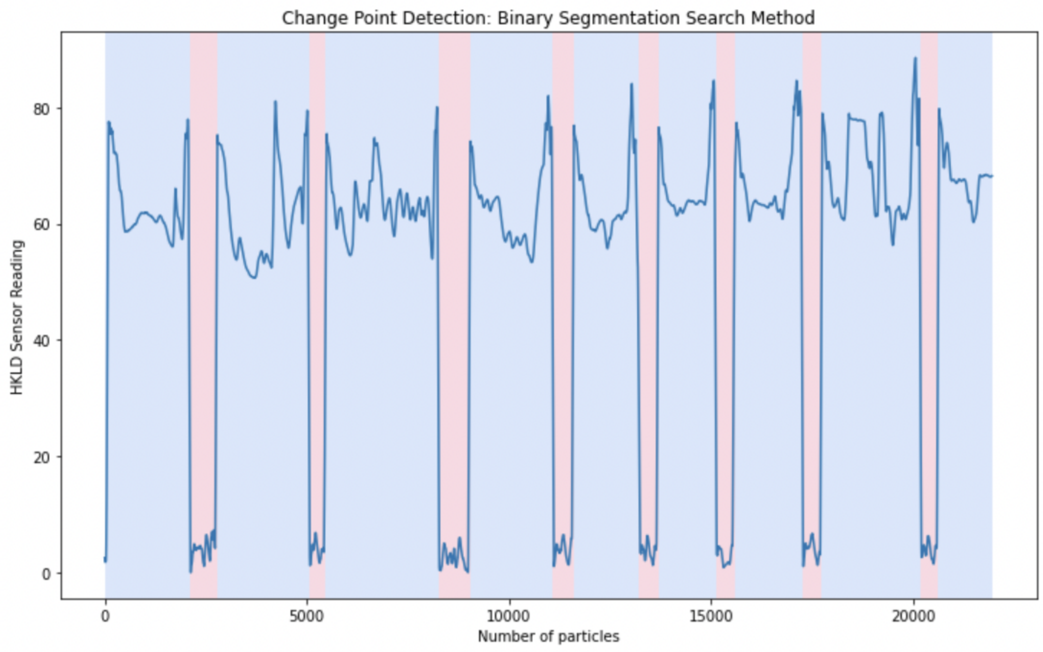


Figure 5.2: Change Point Detection - Binary Segmentation Search Method

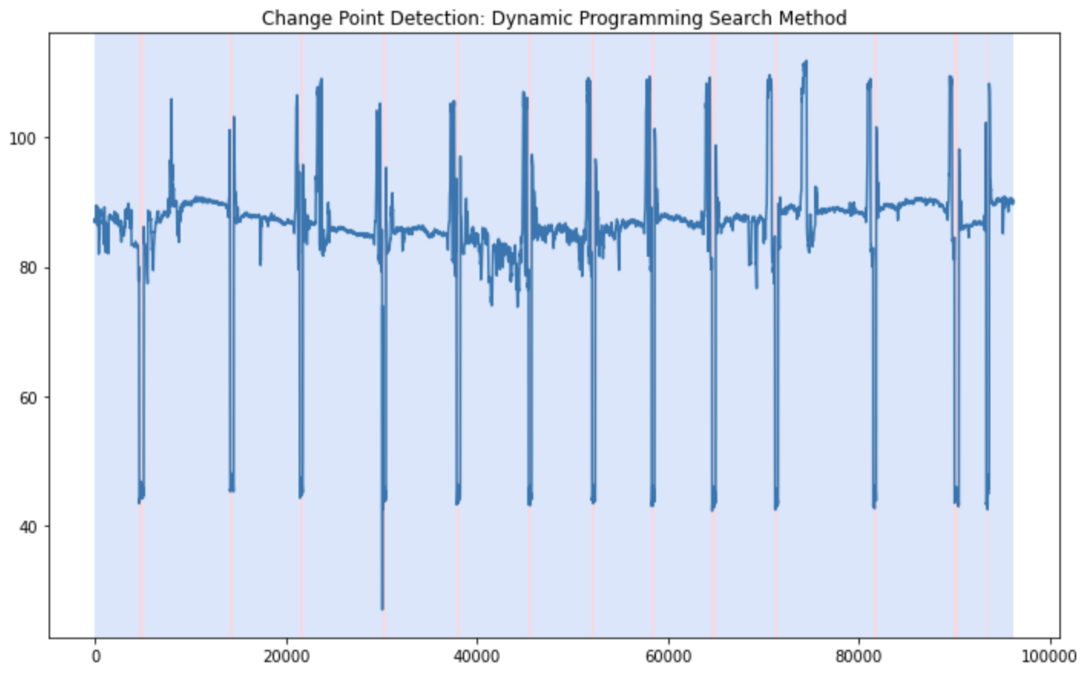


Figure 5.3: Change Point Detection - Dynamic Programming Search Method

Chapter 6

Unsupervised Machine Learning Implementation

6.1 Time Domain Analysis

A time-domain analysis is required prior to implementing Machine Learning at each new stand in order to gain a general understanding of the signals behavior and different features involved.

An example of typical hook load behavior during tripping the drillstring after making up a new connection can be seen in figure 6.1. The block position is also plotted in the same figure to help interpret the course of events. Below is an outline of the course of events:

- The low initial value corresponds to the block's load when it stays still. It corresponds to the weight of the drill string and block itself. Even so, for this signal under time-domain analysis, the hook load value for this rig is somewhat equal to zero, since it is being measured relative to the weight of the drill string and block. In response to the block pulling the drill string downward, the hook load increases rapidly due to static friction.
- During tripping in, the first peak is typically the highest hook load value. This is due because the static friction coefficient is usually higher than the kinetic friction coefficient, so you need more force to get the drill string moving than to keep it rotating. As the drill string moves, the fluid friction increases, resulting in higher shear stress. Also, if the

circulation has been suspended for a while, gel strength may develop before tripping in the string.

- The steady-state peaks and steady-state lows are small fluctuations from the average hook load value. Our average HKLD value equates to the condition where we're tripping, pumps are off, and rotation has not yet begun. Kinetic friction takes over after the first peak and the hook load value drops to the average. Fluctuations are caused by variation in friction and varying conditions in the wellbore. Moreover, the flexibility of the pipe and BHA, which encounters different forces, can also cause fluctuations.

- As shown in the figure below 6.1., the signal will suddenly change once drilling operations are restarted. the drill string experiences acceleration and once the bit touches the bottom of the hole, a reduction in hook load is observed as part of the HKLD is transferred into the formation.

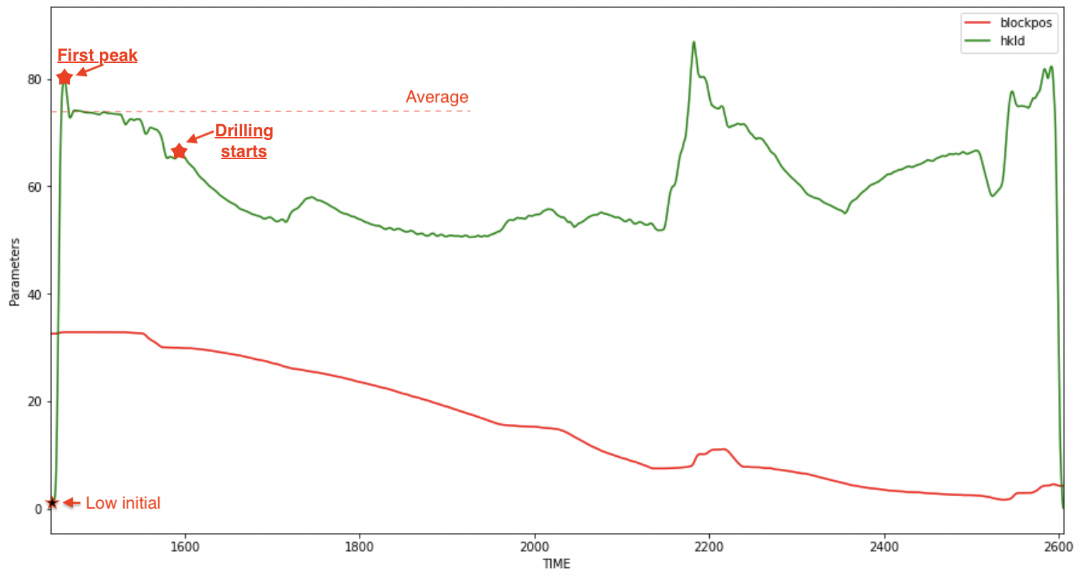


Figure 6.1: Tripping in - Time domain analysis I

Due to varying conditions within the well, it has proven difficult to maintain a constant velocity for the block, as shown in figure 6.2. A correlation between the acceleration of the block and the hook load can be observed over time.

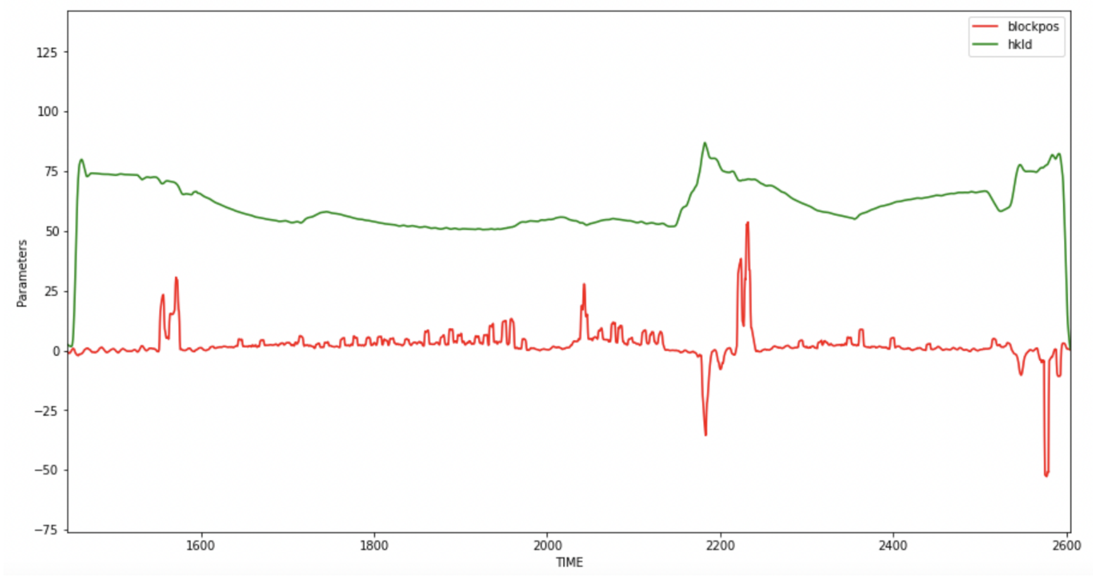


Figure 6.2: Tripping in - Time domain analysis II

6.2 Unsupervised Classification - Clustering

Based on what we described in *Chapter 2*, we will use unsupervised learning on several floating rigs to classify observed behaviors into relevant categories and review recorded data automatically. It is our ultimate goal to cluster the drilling phase from the time we pick up the drillstring until we start tagging the bottom and start drilling. Data that is initially unlabeled will automatically be sorted out by the algorithm based on some hidden features contained in the out-of-slips data set obtained in *Chapter 5*, a set of data that corresponds only to the out-of-slips period.

As a result of grouping the input data, the labeled data will look like this:

- Cluster 1: It is the time from the moment the drillstring is picked up when we are out-of-slips until the time we tag the bottom of the well.
- The remaining clusters correspond to drilling.

6.2.1 Unsupervised Classification - Data Preparation

Since the unsupervised ML algorithm calculates "distance" between points, high-value variables would significantly sway the results. Prior to moving onto unsupervised ML classification, we must first normalize the data set. In other words, feature contributions that contribute more numerically to discriminating pattern classes should be minimized. Then, in order to maintain the same signal level, the features were scaled to unit variance without distorting the ranges of values.

6.2.2 Data Selection

Taking the time to select only those features that are reliable is vital, since every parameter selected will affect the final clustering. A variety of surface and downhole sensors have been used to measure physical phenomena common to the actual environment. Following is the table 6.1, which shows which factors are selected to establish accurate and reliable relationships between variables for the final clusters:

Number	Feature	Units
1	Time	seconds
2	Average Hook load	tonnes
3	Block Velocity	m/minute
4	Block Position	meters
5	Average Surface Torque	kN-m
6	Average Rotary Speed	rpm
7	Pump Pressure	bar
8	Mud Flow In	L/min
9	Equivalent Circulating Density	g/cm3
10	Pump Stroke Count	Dimension-less

Table 6.1: Features selected for clustering

For the well chosen in our study to demonstrate results, a short description of each sensor measurement system is given below:

- Average Hookload: over the top drive, there exists a load cell/strain gage sensor.
- Block velocity: Angular speed using a pair of standard DWE Encoders for redundancy.
- Block position: calculated from the Encoder too. However, a calibration is required for drum size, wire amount and so on.
- Average surface torque: measured from current consumption to top drive.

- Average Rotary speed: belt driven, calculated from an encoder in the gearbox.
- Pump pressure: it is measured from the standpipe with standard Piezoelectric Pressure Sensor.
- Mud flow in: it measures number of strokes vs liner volume, then takes the set up from the DCS to assign the strokes from active pumps to active flow in.
- ECD: calculated from pressure measured downhole (PWD x Strain Gauges) and TVD from the measured survey data.

Due to the following reasons, the following features mentioned in *Chapter 4* were not considered:

- Weight on Bit. Some companies try to calculate WOB in the following way:

  $$\text{WOB} = \text{Tara Hook load } - \text{Hook load Measured} \pm \text{Hook load Correction}$$

  Physical parameters interaction while drilling with the drillstring such as *Q,RPM,Inclination, Azimuth, Cuttings transport, etc.* are used to define the *Hook load Correction* . In our research, we suggest that the *HKLD Correction* can be ignored and dismissed if we are able to find the trend of the free-rotating HKLD for several connections in a row over time during drilling operations. The main reason behind this approach is because currently, there is no model of the well's physical interaction with the drillstring that is accurate, general and straightforward that would allow us to predict the correction factor on the weight on bit. Then, the equation of the WOB calculation using our approach is as follows:

  $$\text{WOB} = \text{Free-Rotating Hook load(At time t, based on the trend found)} - \text{Hook load Measured (t)}$$

  Therefore, the WOB reading in our study will be considered non-optimal and disregarded as a key feature for the clustering. In fact, it should be estimated from our progressive free-rotating HKLD while drilling final result.

- Estimated Rate of Penetration. *ROP* or more correctly called *Estimated ROP*, is measured as a function of the Block velocity sensor reading. It is frequently delayed and innacurate. In light of this, the reading of this parameter is disdained.

- Downhole measurements. Downhole measurements were largely excluded from the beginning, as they were considered unreliable for accurate analysis or mostly unavailable in some cases.

6.2.3 Clustering evaluation - Number of optimal clusters

Making meaningful clusters will be key for the accuracy of our final result. To determine the quantity of clusters we need for our dataset, both domain knowledge-based k value and the Elbow method are used.

To have an initial idea of the approximate number of optimal clusters that should be formed for our dataset, the Elbow method helps us to obtain an optimal number of clusters by fitting a model with a range of *'k'* values.

The number of clusters should be as few as possible to achieve the minimum inertia value. In the graph, we may see that inertia drops at a fast rate at the beginning. As the elbow approaches, there isn't much change in inertia value. Afterwards, the optimal number of clusters*'k'* result from the Elbow method is minimally corrected using domain knowledge to increase accuracy.

6.2.4 Validation

The clustering algorithm is slightly more challenging to validate compared to supervised machine learning algorithms because it lacks ground truth labels. We evaluate the outcome to verify if the model makes meaningful and accurate clusters based on both visual pattern recognition through signal analysis and physics knowledge for every new connection.

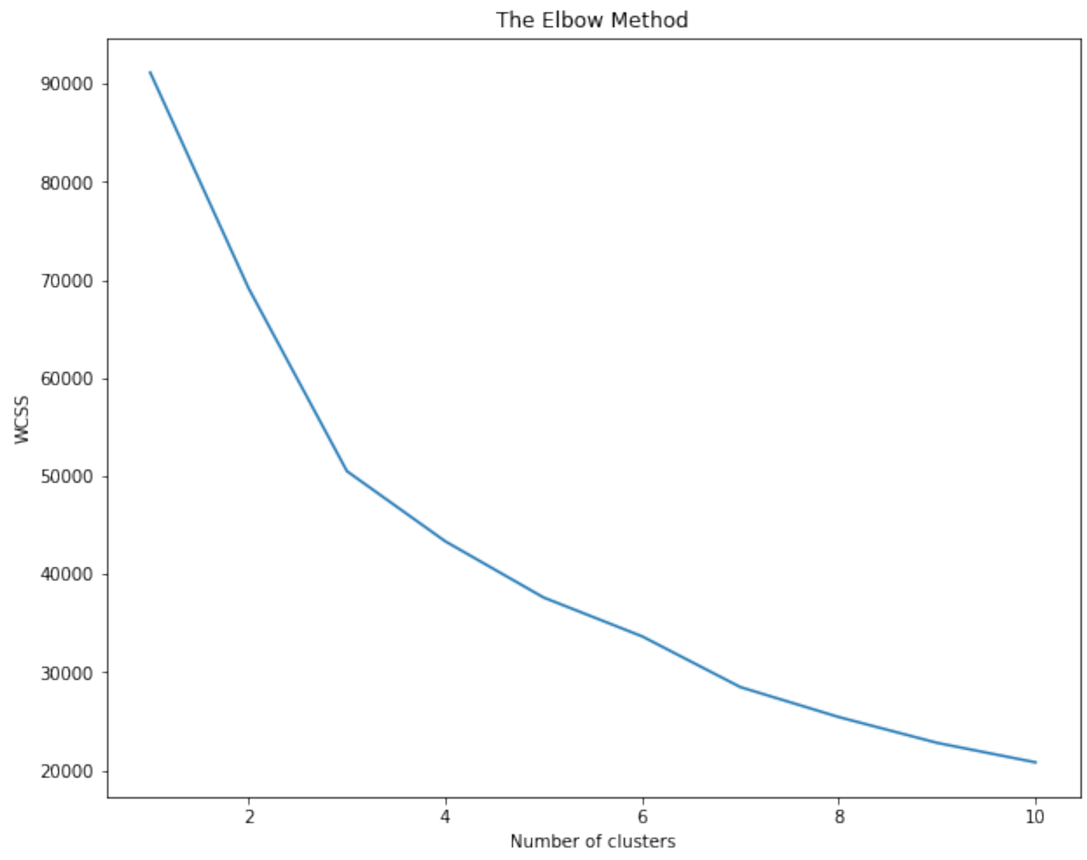


Figure 6.3: Elbow method

When we visualize the physical decision boundaries for every new connection, we can get an intuitive grasp of a learning model's performance and outcome.Based upon the analysis of the datasets, we determine the time intervals where we expect clusters to be detected and compare how each ML method performs.

Expert domain knowledge is needed for this. To ensure the clusters are correct, we evaluate the methods by subjective expectations based on real-time drilling reports and signal behavior data. Also, due to my lack of oil field expertise, external supervisors were consulted for verification of the results and to reassure that the identified areas corresponded to those of interest for every connection.

6.3 Unsupervised ML - Methods

The key objective is to cluster the HKLD behavior for each new connection to assess its applicability to real-time operations and assess the robustness, generality and adaptability of this ML approach.

Having already discussed in *Chapter 2* the three unsupervised classification ML algorithms, we will discuss the results in this subsection. Having chosen the most accurate method, we will proceed with our research.

Below is a visualization of how the HKLD would behave after making up a new stand 6.4; it is followed by an example of HKLD behavior for the entire dataset 6.5.

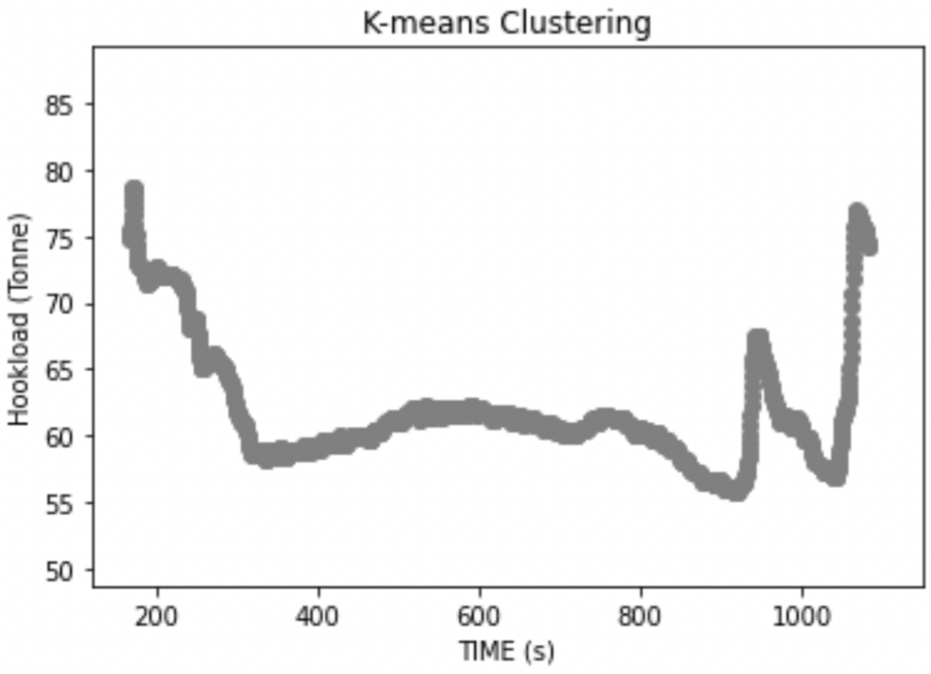


Figure 6.4: HKLD behavior after making up a new connection

As observable, a large amount and variance of hook load data points in the depth range is present for several ran stands. This indication of difficult tripping in conditions is valuable to be present in our unsupervised ML study. In other words, a positive result will ensure robustness

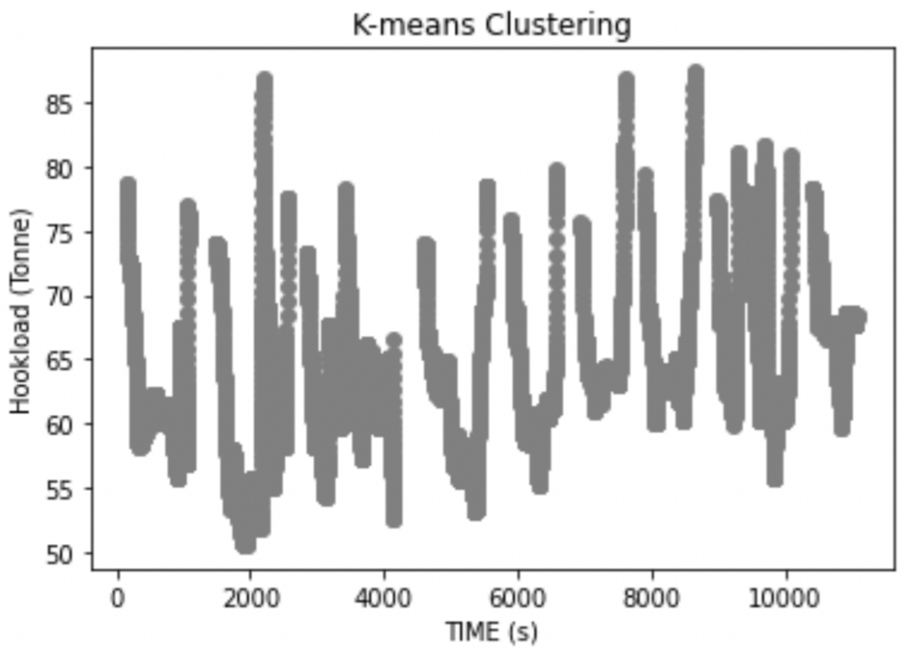


Figure 6.5: HKLD behavior for the entire dataset

as well as generality for its eventual industry application.

Then, the classification algorithm models predict several cluster for each variable simultaneously, where the behavior for every time *t* of each feature selected influences the others within each clustering for a smart and robust cluster detection of the signal of interest *(HKLD)* . To keep the study simple, we will only show the HKLD cluster detection in dependence on the other features selected.

Lastly, we may be able to improve the general clustering result by including the different system states.

Fuzzy c-means

Fuzzy C-means fuzzy is the first unsupervised clustering algorithm tested. As we can see, each data-point simultaneously exists in all clusters to varying degrees of membership from 0 to 1. Nevertheless, soft clustering's membership score allows us to filter out data points that do not belong to a cluster with a high membership.

According to the Elbow Method, the average optimal number of clusters are seven clusters for each new connection for the Fuzzy c-means method.

It is required that the Fuzzy C-means clustering has a "fuzziness" of at least 1. If q = 1.0, we would consider our clustering hard, meaning that the model parameters are restricted to only one cluster. We chose the value q = 2.0 in order to achieve "fuzzy" clustering, which has been proven successful by Hathaway and Bezdek [42]. Our q = 2.0 allows us to have probabilities values for each feature ranging between 0 and 1. As the value gets closer to 1, the greater the likelihood that the parameter is part of that cluster.

Next, after pre-defining the input parameters before mentioned, we will show Fuzzy C-means clustering for the first three new connections in figures 6.6, 6.7 and 6.8.

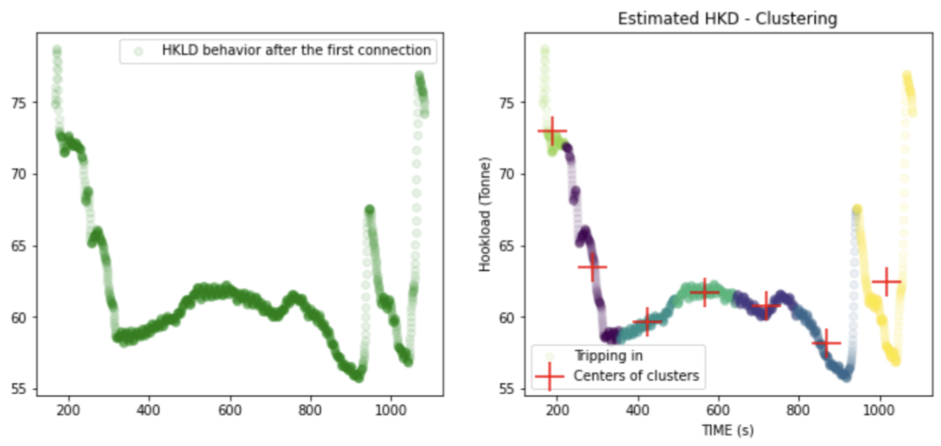


Figure 6.6: Fuzzy c-means Clustering - HKLD behavior after making up first new connection

Based on our validation insight criteria from the HKLD physical interpretation, Fuzzy C-means is not able to detect the whole area of interest corresponding to the first cluster which is overlapped as we can appreciate in the plots with the membership scores of the first and second clusters.

In particular, in the second connection there exists an interval there is a large deviance from our expectations, where we had expected a shorter identification interval than those given by

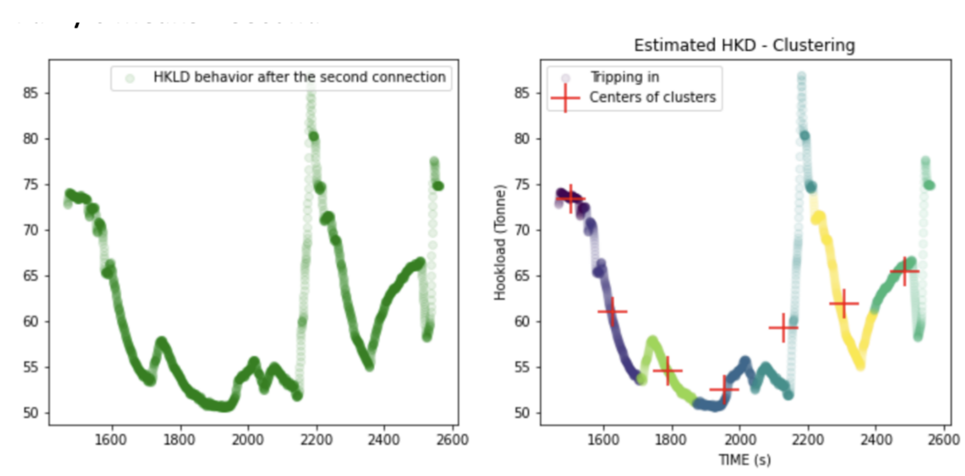


Figure 6.7: Fuzzy c-means Clustering - HKLD behavior after making up second new connection

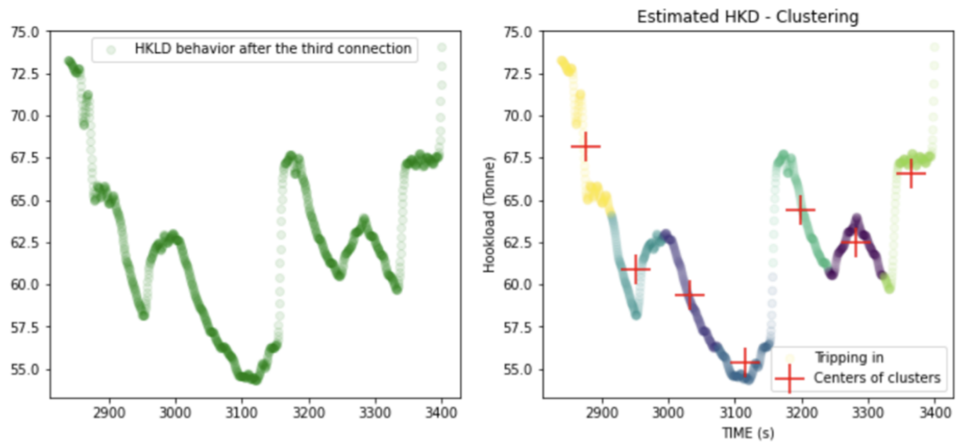


Figure 6.8: Fuzzy c-means Clustering - HKLD behavior after making up third new connection

the Fuzzy c-means methods. Nevertheless, although the overall clustering result of the HKLD signal is promising we need to find a more accurate methodology.

Gaussian Mixture Models (GMM)

GMM technique is implemented in order to classify data according to its probability distribution. We obtain the following results through GMM to cluster each new connection. For simplification, only the clustering for the first connection is shown 6.9.

By fitting a Gaussian Mixture on the HKLD signal, the ML model automatically inferred intrinsic patterns. As we can visualize from the first connection, the inferred intrinsic patterns with similar behavior found are longer than expected. For the first cluster, the HKLD datapoints are expected to be collected within the first 280 seconds of data collection. Definetely

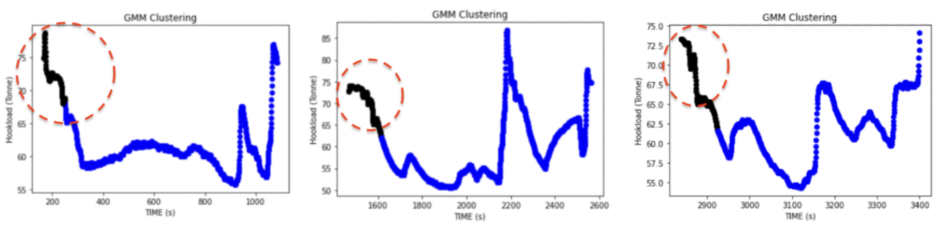


Figure 6.9: GMM Classification - Clustering for the first three connections

this methodology does not lends itself very well to grouping this type of signals.

K-Means clustering

For each new connection, the optimal number of clusters is predetermined and set at eight for the K-means method. The following plot 6.10 shows the step-by-step process of how the algorithm works. The cluster centers are randomly initialized. Every time, data points are assigned to the nearest center, which is updated based on the clustered HKLD signal's relationship to the other features. At last, the HKLD signal datapoints are reassigned to the updated centers. The iteration continues until the centers do not move beyond the tolerance set at 0.05.

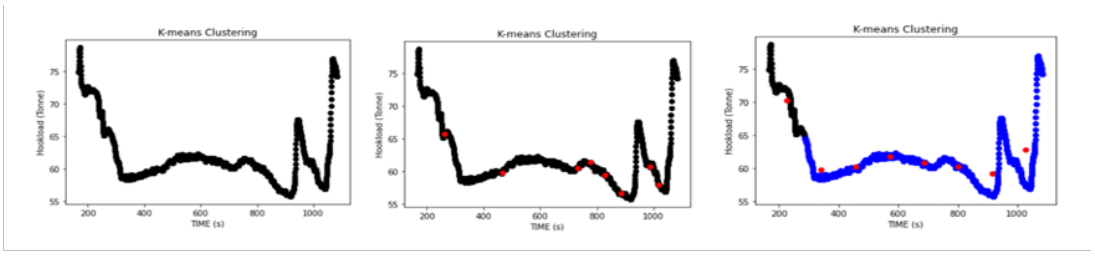


Figure 6.10: K-means Clustering - Step by step clustering

As a result of using K-means to cluster one new connection, we get the following results. For simplification, only the clustering for the first three new connections is shown 6.11, 6.12 and 6.13.

Figure 6.14 below shows the results from the event detection network for all the running in

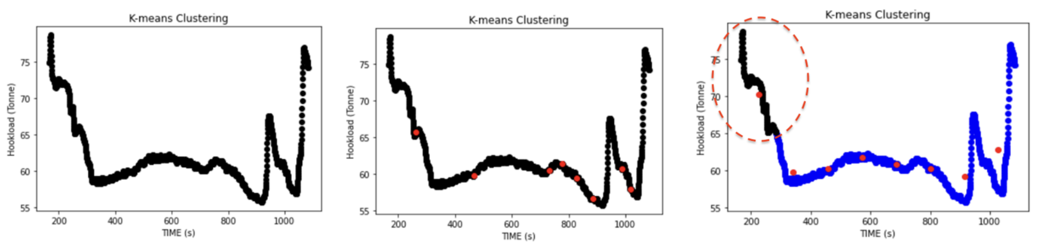


Figure 6.11: K-means Clustering - Clustering for the first new connection

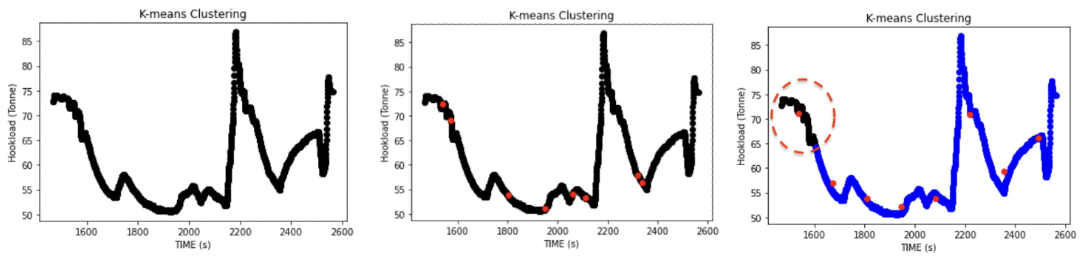


Figure 6.12: K-means Clustering - Clustering for the second connection

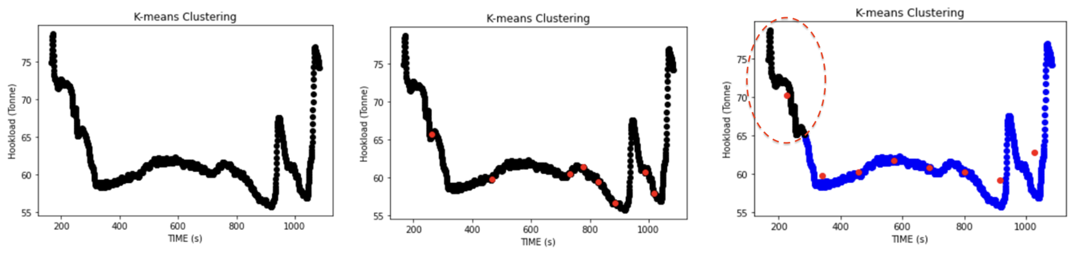


Figure 6.13: K-means Clustering - Clustering for the third connection

hole operations after going out-of-slips through K-means method.

As analyzed from the results, clusters of interest are consistently detected accurately through K-means. As presented in the results, K-means clustering has an excellent potential for detecting non-linear,complex and multivariable engineering behaviors in the HKLD signal.

The performance of the last two clustering methods differ slightly in terms of identifying longer or shorter region if interest window. K-means identify wholly and accurately, all of the regions through hard clustering before tagging bottom. When the bit touches bottom, it has a sharply declining trend, and another cluster is always identified. There exist very little difference between the observed and predicted regions.

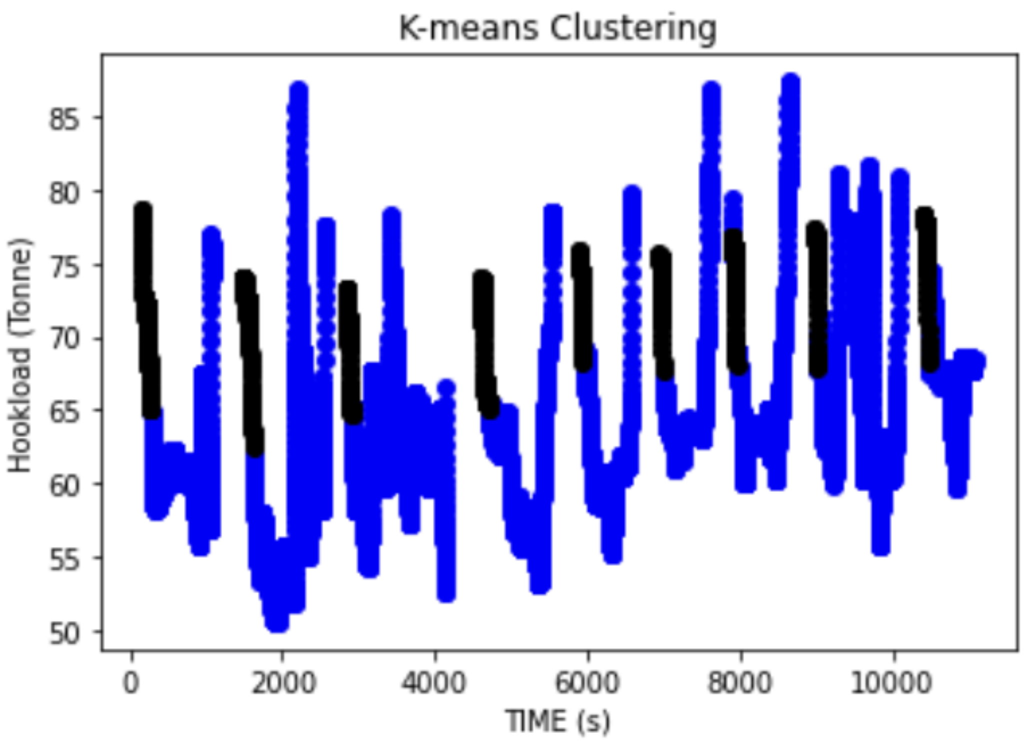


Figure 6.14: K-means Clustering - Classification of the HKLD for the entire dataset

Overall, the results show that K-means method can serve as the best procedure for the classification setting so far, as long as the predefined values are set appropriately. As of now, the methods have only been assessed in terms of subjective expectations. To determine whether the models are satisfactory, experts were consulted as we mention on the Validation subsection presented afterwards.

The performance of each clustering method differs slightly in terms of identifying longer or shorter region of interest window. When the bit touches bottom, it has a sharply declining trend, k-means clustering has the best performance for detecting non-linear, complex, and multivariable engineering behaviors of the HKLD signal of the region of interest for every connection.

Having chosen the most accurate method as k-means, we narrow down the dataset for every connection and proceed with our research.

Chapter 7

Final Results

For each of the nine connections for the rig under analysis, the K-means method has been found to generate the cluster of interest with the highest satisfactory performance as explained in *Chapter 6* in comparison with other approaches. On top of that, this simple but powerful model is more cost-effective and scalable than other complex ones. Again, the identified cluster of interest corresponds with the period of time between picking up weight and approximately the moment we start tagging bottom. Right after narrowing down the dataset to only the cluster of interest *(first cluster)* 7.1, our next step is to catch the optimal free-rotating HKLD for every new make-up connection.

7.1 Calculation of the Optimal Free rotating HKLD

To recreate real-time operations as much as possible from the *Chapter 6* dataset, the data points are constantly fed using a sliding window of five observations which equals to a 2.5 seconds period corresponding to 5 sensor readings in total.

Statistics are used in this section as a tool for the sliding windows of observation to characterize the signals and the processes that generate them [43]. For the purpose of analyzing the signal in this part of our study, we are particularly interested in the mean and the variance. Variance informs the user about the variability of the sliding window, and is used to detect mainly

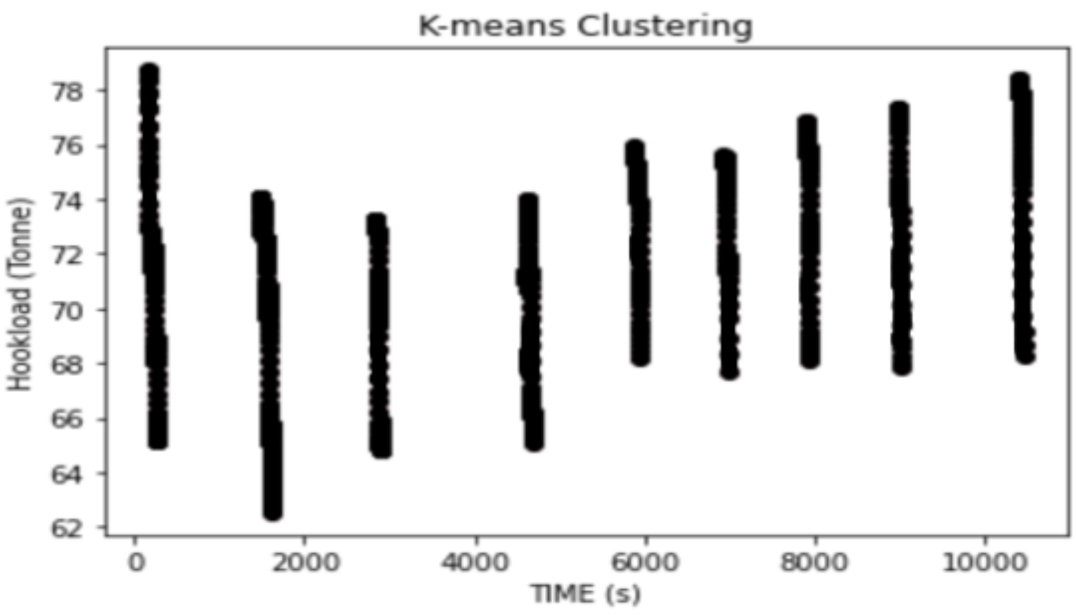


Figure 7.1: K-means Clustering - identified cluster for every connection

the steady state region of the features.

A running tally of every feature is kept as the signal moves through:

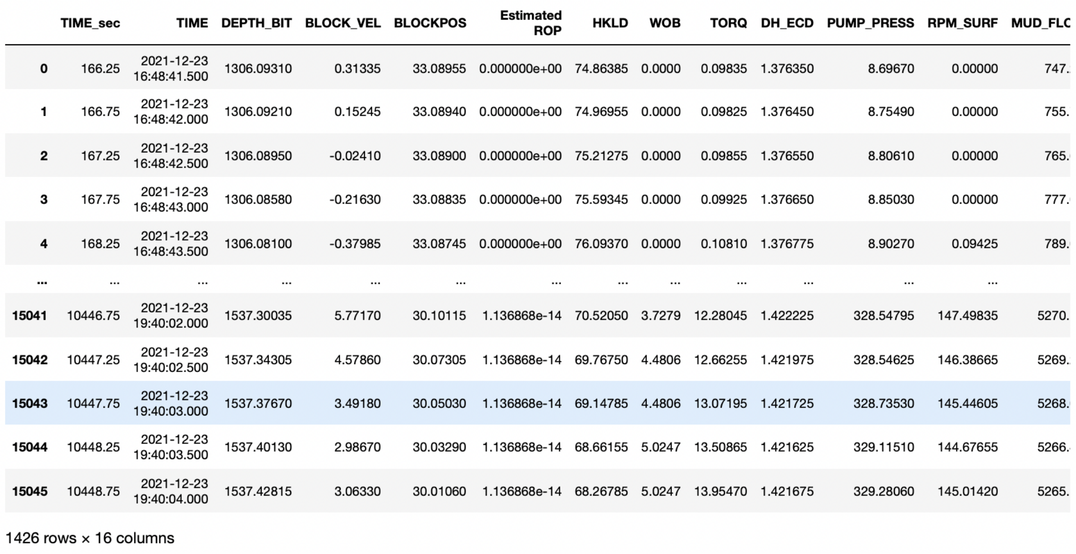
- The index and value.
- The variance of the sliding window.
- The mean of the sliding window.

In data analysis, running mean and variance creates a series of averages and variances respectively from different samples of the dataset. Taking a series of samples of a predetermined size, the first element in the running mean and variance is calculated by averaging the samples and calculating the variance over a predetermined size. After that, the initial sample is modified by shifting forward, followed by overwriting the first element for the next index value of the feature as in real time operations.

In a continuous data stream, we aim to detect steady state regions from the mean and variance corresponding to several features. In simple terms, stationarity means that the statistics of the signals are constant over time (i.e. mean and variance).

7.1.1 Methodology

To start with, we split the resulting dataframe from *Chapter 6* into multiple dataframes based on jumps in the index sequence so as to discern a connection from the next, as presented in the next figure 7.2.



	TIME_sec	TIME	DEPTH_BIT	BLOCK_VEL	BLOCKPOS	Estimated ROP	HKLD	WOB	TORQ	DH_ECD	PUMP_PRESS	RPM_SURF	MUD_FLC
0	166.25	2021-12-23 16:48:41.500	1306.09310	0.31335	33.08955	0.000000e+00	74.86385	0.0000	0.09835	1.376350	8.69670	0.00000	747.
1	166.75	2021-12-23 16:48:42.000	1306.09210	0.15245	33.08940	0.000000e+00	74.96955	0.0000	0.09825	1.376450	8.75490	0.00000	755.
2	167.25	2021-12-23 16:48:42.500	1306.08950	-0.02410	33.08900	0.000000e+00	75.21275	0.0000	0.09855	1.376550	8.80610	0.00000	765.
3	167.75	2021-12-23 16:48:43.000	1306.08580	-0.21630	33.08835	0.000000e+00	75.59345	0.0000	0.09925	1.376650	8.85030	0.00000	777.
4	168.25	2021-12-23 16:48:43.500	1306.08100	-0.37985	33.08745	0.000000e+00	76.09370	0.0000	0.10810	1.376775	8.90270	0.09425	789.
...	...	...	...	...	...	...	...	...	...	...	...	...	
15041	10446.75	2021-12-23 19:40:02.000	1537.30035	5.77170	30.10115	1.136868e-14	70.52050	3.7279	12.28045	1.422225	328.54795	147.49835	5270.
15042	10447.25	2021-12-23 19:40:02.500	1537.34305	4.57860	30.07305	1.136868e-14	69.76750	4.4806	12.66255	1.421975	328.54625	146.38665	5269.
15043	10447.75	2021-12-23 19:40:03.000	1537.37670	3.49180	30.05030	1.136868e-14	69.14785	4.4806	13.07195	1.421725	328.73530	145.44605	5268.
15044	10448.25	2021-12-23 19:40:03.500	1537.40130	2.98670	30.03290	1.136868e-14	68.66155	5.0247	13.50865	1.421625	329.11510	144.67655	5266.
15045	10448.75	2021-12-23 19:40:04.000	1537.42815	3.06330	30.01060	1.136868e-14	68.26785	5.0247	13.95470	1.421675	329.28060	145.01420	5265.

1426 rows × 16 columns

Figure 7.2: Running in hole Dataframe

Thereafter we conduct statistical analysis in streaming setting for each new connection. For every connection we may distinguish the following patterns:

- Tripping in.

- Rotating Off-Bottom: rotary speed and pump rate must be activated. Also, bit depth must be almost constant and be off-bottom.

The main idea is at first to walk through the signal with a window of a fixed size. For each step, a function computes the before-mentioned tally of every feature under consideration. For a row of features to be saved and identified as in free-rotating condition, all the pre-set statistical and physical thresholds for every following feature must be met:

- Stand pipe pressure *SPP* and Mud flow Q maximum variance: to ensure that the flow is active and steady so that the free-rotating HKLD resembles drilling conditions as closely as possible.
- Bit depth maximum standard deviation. The bit depth must be almost constant to comply with the rotating off-bottom conditions.
- RPM Maximum Variance: to ensure that the rotary speed does not fluctuate excessively.
- RPM Mean: for the current trip interval, the rotation speed must be greater than a reference value selected by the user.

After meeting all the pre-set statistical and physical thresholds for the features under analysis, the free-rotating region is detected as shown in green in the figure 7.3.

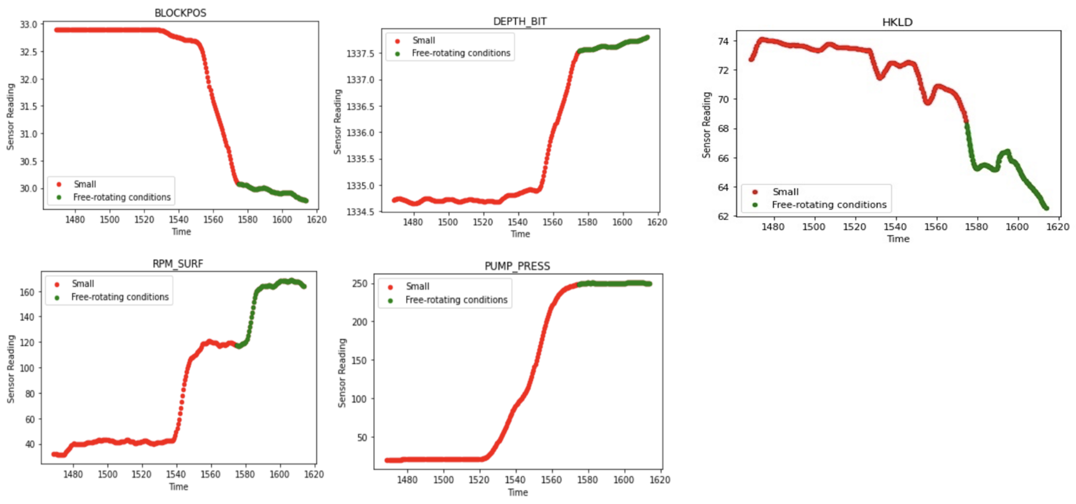


Figure 7.3: Free-rotating Region Detection

After identifying this region, we are then able to calculate the optimal HKLD. The mean of every new connection for each region is calculated and considered as the optimal HKLD. The next graph 7.4 shows the optimal off-bottom HKLD for every connection.

7.2 Trend

The final objective is to find a way to update the free-rotating hookload while drilling to adapt it to the changing drilling conditions (inclination, azimuth, flow rate ...) as compared to the initial

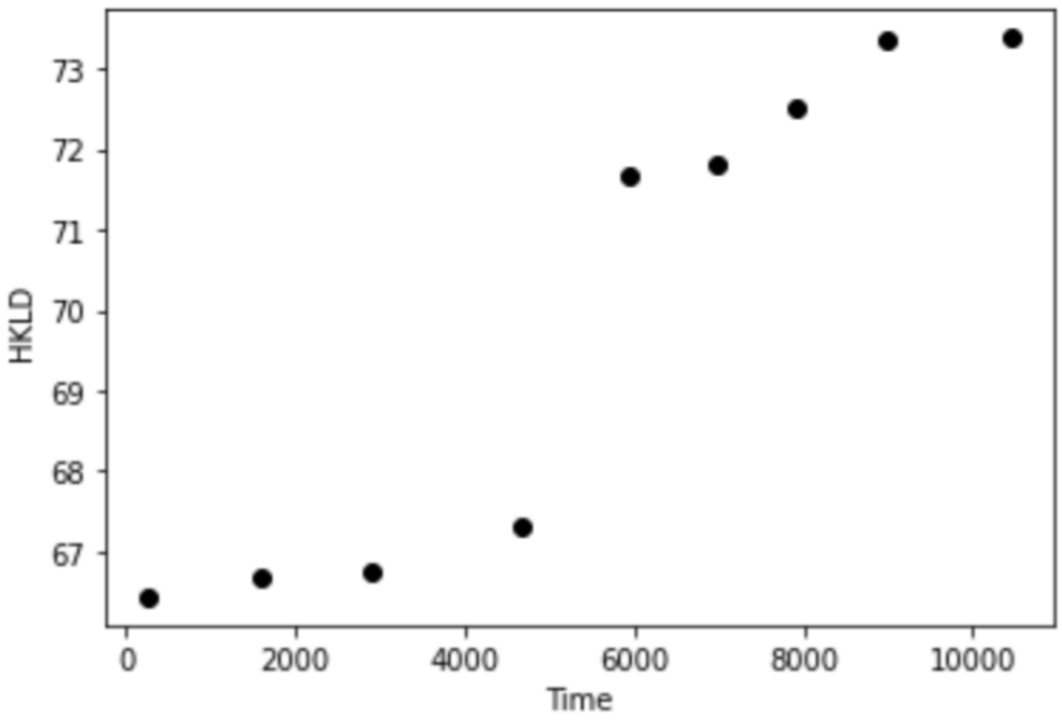


Figure 7.4: Free-rotating HKLD of every connection

conditions where we measured the free-rotating HKLD.

To have confidence in the trend determined to be used for updating the free rotating HKLD to the drilling conditions once drilling has been re-started, it is very important for drilling operations not to change suddenly from one connection to another. We observed this clearly in our investigation when the driller entered into a fragile formation area. This critical remark was reported in the drilling report and also reflected in our dataset. In particular, during drilling operations, after making-up the fourth connection, a fragile formation area of the Balder Formation composed of laminated fissile shales with interbedded volcanic tuffs was detected. This caused it to be decided to increase suddenly the pump stroke count, decrease the ECD for subsequent running in holes and subsequently a Stand Pipe Pressure drop. The result is an average free rotating HKLD value that is approximately 10% greater for the new connection that what was previously calculated. Therefore, a new trend for new drilling conditions should be formed as shown in figure 7.6 and ignore the latter due to the wide variance of drilling conditions from one connection to another.

An additional challenge relates to how statistics change over time. As a consequence, our pre-set statistical threshold, which is utilized to calculate the off-bottom HKLD during free rotation, becomes inaccurate over time, negatively affecting our final result. This could lead to a several hour long window where the predictive model is unable to estimate the trend accurately until it is updated by domain experts. Despite this, we show that our predictive model is maintaining a high prediction accuracy after the first connection for new drillstring make-ups. The main reason for this is that the system is tested for only 3 hours for part of the 16-inch-diameter drilled section. Though we recognize the problem, we have yet to find an appealing solution to adequately address it.

An advantage, however, is that this expertise is most needed in the development phase of these models, whereas maintaining the models updated is not as challenging and time-consuming.

Through a Linear Regression approach [44] to modelling the relationship between the optimal HKLD points and time, a trend is found. A new linear trend is found for every new recorded free-rotating HKLD value for every new connection. As can be observed in the plot 7.4, the trend starts to become robust and reliable from the third connection, which means after the third optimal free-rotating HKLD valued is estimated:

A new trend for new drilling conditions is obtained for the so-called drilling phase II 7.6 which becomes more and more robust to be applicable for new connections after the third connection too.

As a result, both figures 7.7 and 7.8 depict the final trend in a clear and simple way.

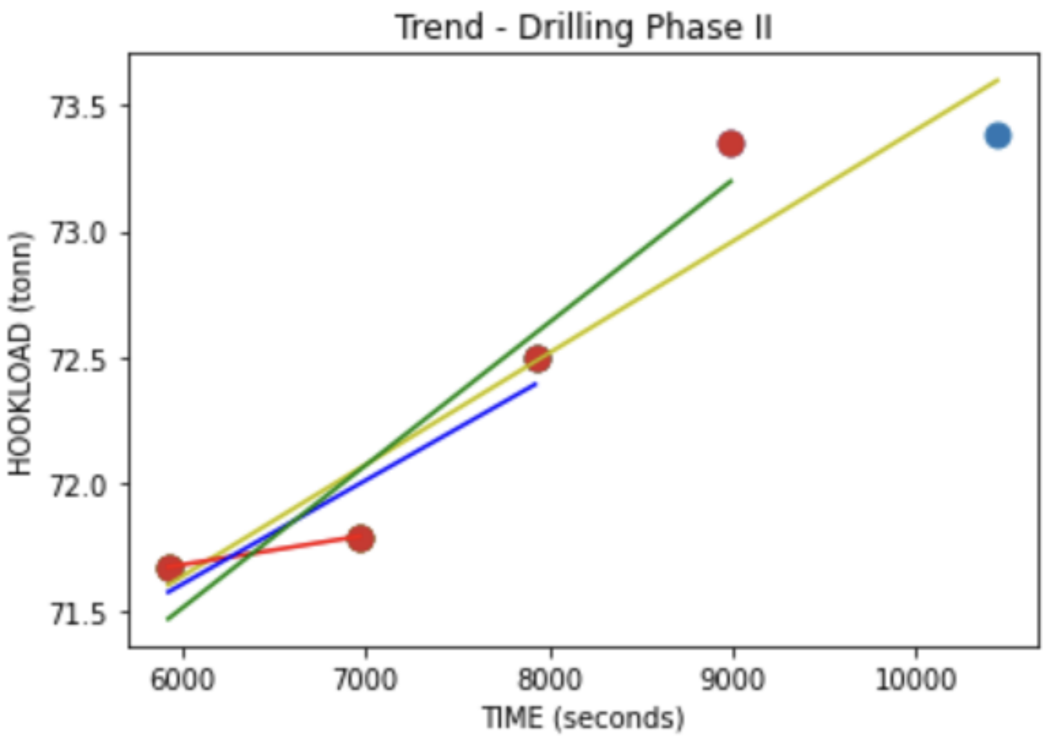


Figure 7.5: Change of the Free-rotating HKLD Trend for the Drilling Phase I

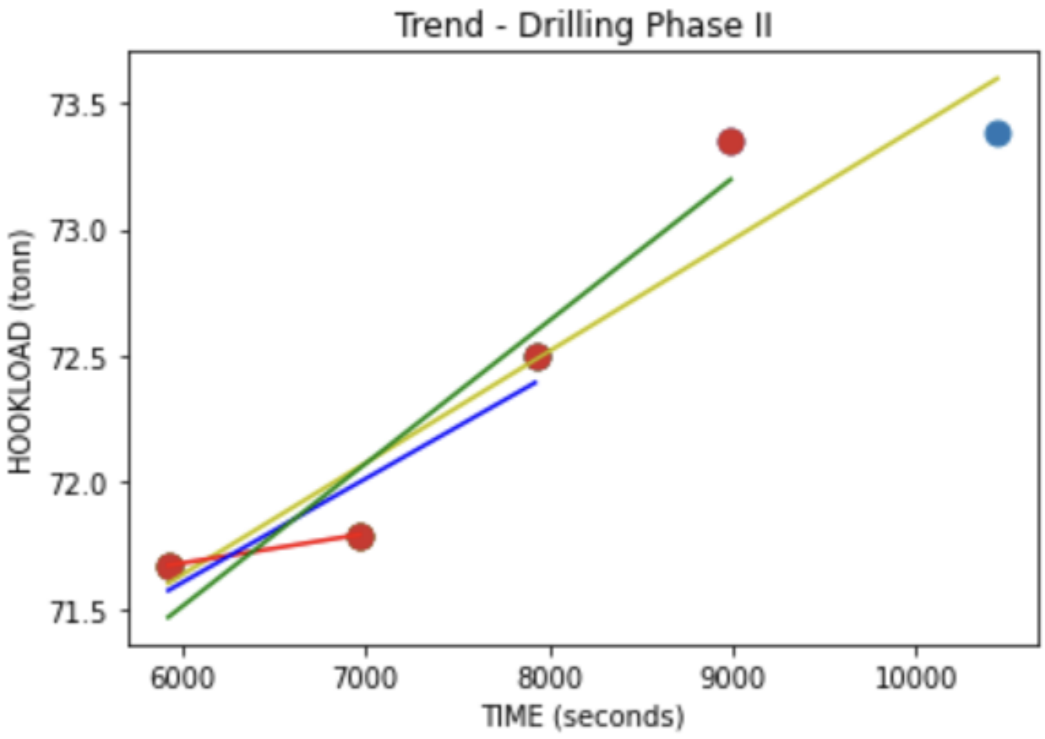


Figure 7.6: Change of the Free-rotating HKLD Trend for the Drilling Phase II

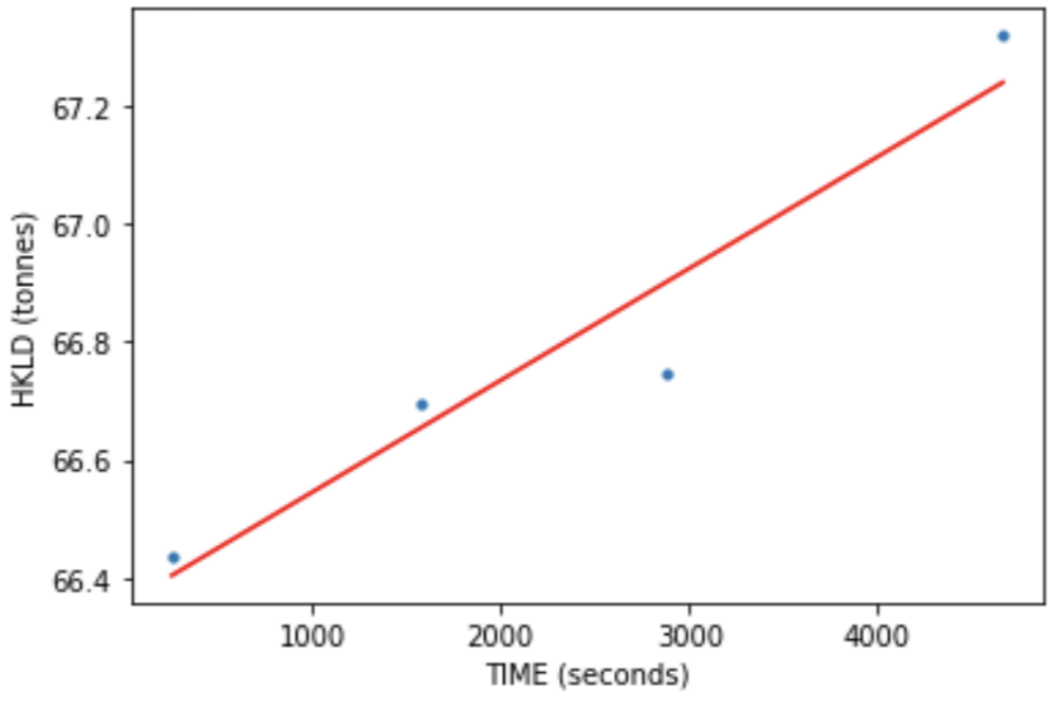


Figure 7.7: Final Free-rotating HKLD trend for the Drilling Phase I

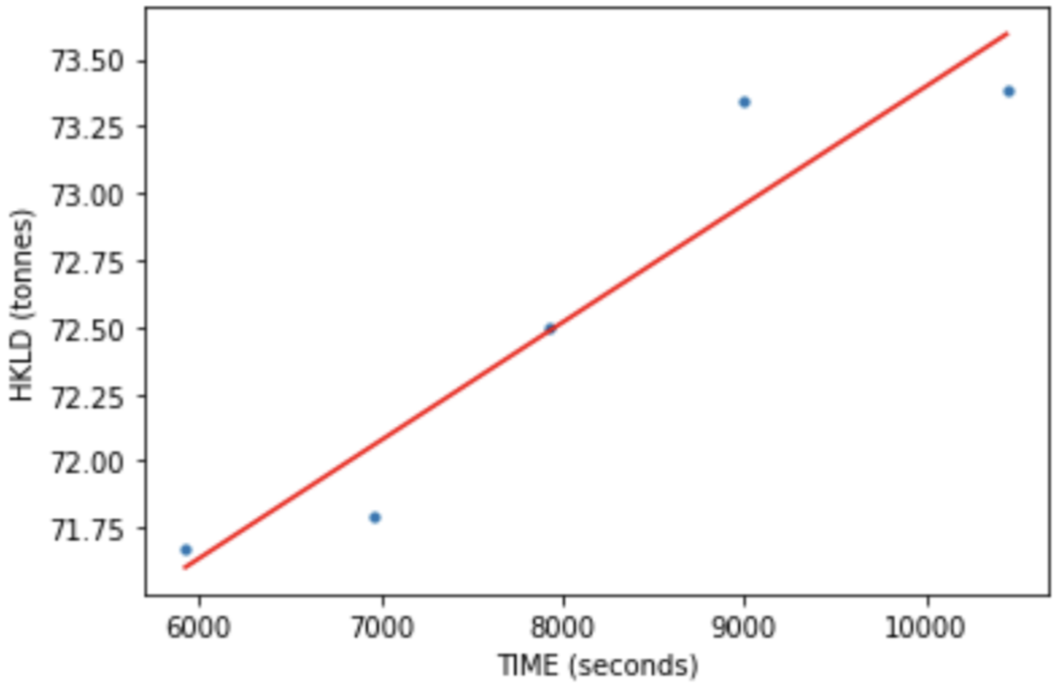


Figure 7.8: Final Free-rotating HKLD trend for the Drilling Phase II

Chapter 8

Conclusion and Future Work

8.1 Conclusions

The traditional method of selecting free-rotating free-stop HKLD by hand is inefficient and inaccurate. This methodology makes it possible to implement automatic, fast, reliable, and manageable free-rotation detection for subsequent runs in hole.

In addition, the rig state engine developed automatically classifies the HKLD signal as out-of-slips, period between pick-up of the drillstring and reaching bottomhole, free-rotating region as well as drilling period. The prediction results have been validated through every new connection. Based on our findings, the methods chosen generated clusters in the expected areas, had a high level of performance, and were low in computational complexity.

Based on the results of implementing a variety of different methods of different types and designs, we concluded that the best performance was achieved by implementing *Binary Segmentation Method* for in-slips detection, *K-means clustering* for running in hole detection, until we tag bottom using the history of sensor data to make its prediction. In the last step, there exists a defined statistical set-up to capture the free-rotating region for every new connection.

Even though we are capable of providing outstanding results, it is imperative to be proficient in machine learning and have technical skills. Thus, the combination of these two skills is a challenge, since engineers must understand the problem, while data scientists need to implement the technical real-time implementation.

In conclusion, we examined many hypotheses regarding the potential of using different approaches to improve the models performance during the planning and execution stages of our experiment, and we set objectives to achieve this goal. Our conclusions are:

- Our study utilized a complete set of well data, parameters were selected from over 30 features, and the data was cleaned and prepared for further analysis.
- An algorithm that detects out-of-slips through pattern recognition was successfully implemented.
- The most important parameters for the ML predictive models were identified. A variety of unsupervised ML algorithms were tested, but K-means performed the best.
- Using the embedded rig state identification engine, different real-time data regions can be classified (e.g., out-of-slips, pick-up, rotating off-bottom and drilling) and used in other applications such as T&D calibration.
- Rather than treating this only as a "data science" problem, petroleum engineering knowledge is used to improve the quality of model inputs, thus enhancing the performance of the model. The use of machine learning as a substitute for domain knowledge is counterproductive for this research. In fact, we may conclude that the most effective AI applications for industry will combine domain expertise with AI tools.

8.2 Future Work

As a result of this study, better WOB estimation solutions can be developed, but before doing so, it is recommended that the following steps be followed:

- More extensive testing, including wells with varying configurations that represent different operational scenarios (sections, inclinations, rig types) is required.

- Test and adapt the hybrid model workflow to real-time field data to ensure its application to the industry to provide fast, efficient, and accurate assessment of incoming data in real-time.

- HKLD sensor is placed on top of the drive sub for the specific rig on which we demonstrated the results. This means that it is not necessary to consider some of the loads that could distort the real from the recorded weight of the drill string, in this case, the sheave friction effect, weight of the traveling block, and standpipe pressure effect. However, HKLD on other rigs under study may be indirectly measured and may require a correction, such as mud hose weight effect, sheave angle, sheave efficiency, and others which have not been evaluated in this study.

- Using the dynamic model developed to detect when the bit is on or off bottom, it is also possible to suggest to the driller a procedure for starting the drilling. It would be possible, for example, to mitigate vibrations to start drilling with the least amount of axial vibration possible.

- It is crucial to have a fully synchronized real-time data source. Desynchronized timestamps affect the accuracy of machine learning models that require precise timestamps within a narrow window to detect, alert and act. A practical solution should be provided to detect and resolve the time synchronization issue between sensors located downhole and others. By identifying the drift in each data source and correcting the data based on its trends, it would be possible to correct the data for the time drift, and rely more on downhole measurements for this study.

References

[1] Miguel Fernandez Berrocal, Jie Cao, and Dan Sui. Evaluation and interpretation on data driven rop models from engineering perspectives, June 2022. ASME 2022 41th International Conference on Ocean, Offshore and Arctic Engineering.

[2] A.T. Bourgoyne, K.K. Millheim, M.E. Chenevert, and F.S. Young. *Applied Drilling Engineering*. SPE Textbook Series, 1991.

[3] Pérez F. Granger B. Bussonnier M. Frederic J. Kelley K. Hamrick J. Grout J. Corlay S. Ivanov P. Avila D. Kluyver T., Ragan-Kelley B. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90, 2016.

[4] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[5] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, USA, 3 edition, 2007.

[6] G. R. Luke and H. C. Juvkam-Wold. The determination of true hook-and-line tension under dynamic conditions. *SPE Drilling & Completion*, 8(04):259–264, Dec 1993. SPE.

[7] Skalle P. Johansen S.T. et al. Mme, U. Analysis and modeling of normal hook load response during tripping operations, 2012.

[8] Eric Cayeux, Hans Joakim Skadsem, and Roald Kluge. Accuracy and correction of hook load measurements during drilling operations, Mar 2015. SPE.

[9] Robert F. Mitchell and Stefan Z. Miska. *Fundamentals of Drilling Engineering*. SPE Textbook Series, 2011.

[10] Z. Wu, G. Hareland, S. M. Loggins, S. Lai, A. Eddy, and L. Olesen. A new method of calculating rig parameters and its application in downhole weight on bit automation in horizontal drilling, Apr 2017. SPE.

[11] Åge Kyllingstad and K. Erik Thoresen. Improving surface wob accuracy. 2018.

[12] R. Teale. The concept of specific energy in rock drilling. *International Journal of Rock Mechanics and Mining Sciences Geomechanics Abstracts*, 2(1):57 – 73, 1965.

[13] C.A. Johancsik, D.B. Friesen, and Rapier Dawson. Torque and drag in directional wells-prediction and measurement. 1984.

[14] T. Eren and M. E. Ozbayoglu. Real time optimization of drilling parameters during drilling operations. In *In SPE Oil and Gas India Conference and Exhibition.*, (2010).

[15] David A. Elley, Norbert Meierhoefer, and Michael S. Strathman. Time-based real time drilling operations excellence delivered, Jan 2007. SPE.

[16] W. G. Lesso, H. Laastad, A. Newton, and T. S. Olberg. The utilization of the massive amount of real time data acquired in wired-drillpipe operations. 2008.

[17] Geir Hareland, Andrew Wu, and Lingyun Lei. The field tests for measurement of downhole weight on bit(dwob) and the calibration of a real-time dwob model, Jan 2014. IPTC.

[18] B. Mantha and R. Samuel. Rop optimization using artificial intelligence techniques with statistical regression coupling. 2016.

[19] C.I. Noshi and J.J. Schubert. Application of data science and machine learning algorithms for rop prediction: Turning data into knowledge. 2019.

[20] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics 32-57*, 1973.

[21] J. C. Bezdek. Pattern recognition with fuzzy objective function algoritms. *Plenum Press*, 1981.

[22] J. B. MacQueen. Some methods for classification and analysis of multivariate observations, proceedings of 5-th berkeley symposium on mathematical statistics and probability. *University of California Press, 1:281-297*, 1967.

[23] Reynolds D. Gaussian mixture models. *Encyclopedia of Biometrics*, 2009.

[24] Aamo O. Strecker.T. Attenuating heave-induced pressure oscillations in offshore drilling by downhole flow control. *IFAC Journal of Systems and Control*, 8, 2019.

[25] K. Gjerstad, S. Tore Thorsen, and R. Bergerud. Exploiting the full potential in automated drilling control by increased data exchange and multi disciplinary collaboration. 2020.

[26] A. Rahman Mohammad Sultan Rasel Hassan Ibrahim Khaled, Mohamed Hasan. Cfd approach to investigate the effect of mud rheology oncuttings removal in horizontal wells. *American Association of Drilling Engineering*, 2020.

[27] Ahmed A. Elgibaly and Mohammed Shehata Farhat. A study of friction factor model for directional wells. *Egyptian Journal of Petroleum*, 26, 2017.

[28] Stefan Yu Mengjiao Takach Nicholas Ahmed Ramadan Zettner Claudia Duan, Mingqin Miska. Transport of small cuttings in extended reach drilling. *SPE Drilling Completion*, 2013.

[29] Ahmed A. Elgibaly, Mohammed Shehata Farhat, Eric W. Trant, and Mohammed Kelany. A study of friction factor model for directional wells. *Egyptian Journal of Petroleum*, 26(2):489–504, Jun 2017.

[30] R. F. Mitchell and S. Z. Miska. Fundamentals of drilling engineering. Society of Petroleum Engineers, (2010).

[31] A.T. Bourgoyne and F.S. Young. A multiple regression approach to optimal drilling and abnormal pressure detection. 1974.

[32] Robello Samuel. Friction factors: What are they for torque, drag, vibration, bottom hole assembly and transient surge/swab analyses? *Journal of Petroleum Science and Engineering*, 73(3):258–266, Sep 2010.

[33] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[34] Gerhard Wallnoefer Gerhard Ettl Johannes Mathis, Wolfgang Thonhauser. Use of real-time rig-sensor data to improve daily drilling reporting, benchmarking, and planning. *SPE Drilling Completion*, 2007.

[35] Bilal Fruhwirth Rudolf Thonhauser Gerhard Esmael Bilal Prohaska Michael Arnaout, Arghad Alsallakh. Diagnosing drilling problems using visual analytics of sensors measurements. *SPE Drilling Completion*, 2012.

[36] Mohd Mustafa Al Bakri Abdullah. Filling missing data using interpolation methods: Study on the effect of fitting distribution. *Key Engineering Materials*, 594-595:889–895, 01 2014.

[37] Dan Sui. *Drilling Automation and Modeling Compendium*, pages 171–172. 2019.

[38] Sebastian Raschka, David Julian, and John Hearty. *Python: Deeper Insights into Machine Learning*, pages 112–113. Packt Publishing, 2016.

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

[40] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media, Inc., 2017.

[41] A. Tunkiel, D. Sui, and T. Wiktorski. Reference dataset for rate of penetration benchmarking. *Journal of Petroleum Science and Engineering.*, 2020. doi: 10.1016/j.petrol.2020.108069.

[42] R.J. Hathaway and J.C. Bezdek. Fuzzy c-means clustering of incomplete data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 31(5):735–744, 2001.

[43] Ronald Deep. *Probability and Statistics with Integrated Software Routines*, page 290. Academic Press, 2005.

[44] Gary Smith. *10 - Multiple Regression*, pages 301–337. Academic Press, Boston, Jan 2015.

Appendices

Appendix A

Python Code

A.1 Installed Packages

Name	Version	Name	Version
absl	1.0.0	appnope	0.1.2
argon2-cffi	21.3.0	argon2-cffi-bindings	21.2.0
asgiref	3.5.0	asttokens	2.0.5
attrs	21.4.0	autograd	1.4
backcall	0.2.0	backports.functools-lru-cache	1.6.4
backports.tempfile	1.0.0	backports.weakref	1.0
beautifulsoup4	4.10.0	bleach	4.1.0
brotlipy	0.7.0	certifi	2021.10.8
cffi	1.15.0	changefinder	0.3
chardet	4.0.0	charset-normalizer	2.0.4
click	8.0.4	cma	2.7.0
colorama	0.4.4	conda	4.12.0
conda-package-handling	1.8.1	conda-build	3.21.8
conda-verify	3.4.2	cryptography	36.0.0
cycler	0.11.0	debugpy	1.5.1
decorator	5.1.1	defusedxml	0.7.1
Django	1.0.0	eia-python	1.22
entrypoints	0.3	et-xmlfile	1.1.0
executing	0.8.3	fastjsonschema	2.15.1
filelock	3.6.0	fonttools	4.32.0
future	0.18.2	fuzzy-c-means	1.6.4
glob2	0.7	HeapDict	1.0.1
idna	3.3	image	1.5.33
importlib-metadata	4.11.3	ipykernel	6.9.1
ipython	8.2.0	ipython-genutils	0.2.0
jedi	0.18.1	Jinja2	2.11.3
jupyter-core	4.9.2	jupyterlab-pygments	0.1.2
kiwisolver	1.4.2	libarchive-c	2.9
MarkupSafe	2.0.1	matplotlib	3.5.1
matplotlib-inline	0.1.2	mistune	0.8.4
mpmath	1.2.1	navigator-updater	0.2.1
nbclient	0.5.11	nbconvert	6.4.4

nbformat	5.3.0	nest-asyncio	1.5.5
notebook	6.4.8	numpy	1.22.3
openpyxl	3.0.10	ortools	9.3.10497
packaging	21.3	pandas	1.4.2
pandocfilters	1.5.0	parso	0.8.3
patsy	0.5.2	pexpect	4.8.0
pickleshare	0.7.5	Pillow	9.1.0
pip	22.0.4	pkginfo	1.8.2
plotly	5.6.0	prometheus-client	0.13.1
prompt-toolkit	3.0.20	protobuf	3.20.0
psutil	5.8.0	ptyprocess	0.7.0
pure-eval	0.2.2	pycosat	0.6.3
pycparser	2.21	pydantic	1.9.0
Pygments	2.11.2	pymoo	0.5.0
pyOpenSSL	22.0.0	pyparsing	3.0.4
pyrsistent	0.18.0	PySocks	1.7.1
pyswarms	1.3.0	python-dateutil	2.8.2
pytz	2021.3	PyYAML	6.0
pyzmq	22.3.0	QtPy	2.0.1
requests	2.27.1	ruamel-yaml-conda	0.15.100
ruptures	1.1.6	scikit-learn	1.0.2
scipy	1.8.0	seaborn	0.11.2
Send2Trash	1.8.0	setuptools	62.1.0
sip	4.19.13	six	1.16.0
sko	0.5.7	soupsieve	2.3.1
sqlparse	0.4.2	stack-data	0.2.0
statsmodels	0.13.2	streamz	0.6.3
sympy	1.10.1	tabulate	0.8.9
tabulate	0.8.9	tenacity	8.0.1
terminado	0.13.1	testpath	0.5.0
threadpoolctl	3.1.0	toolz	0.11.2
tornado	6.1	tqdm	4.63.0
traitlets	5.1.1	typer	0.4.1
typing_extensions	4.1.1	urllib3	1.26.8
wcwidth	0.2.5	webencodings	0.5.1
wheel	0.37.1	typer	0.4.1
zict	2.2.0	zipp	3.7.0

A.2 Data Cleaning Code

```python
 #change df type from object to float64 except for TIME=datetime64
[ns]
 for col in  df.columns[1:]:
     df[col] = pd.to_numeric(df[col], errors='coerce')

 #Historic data
 numerical_attributes= df.select_dtypes(include='float')
 numerical_attributes.hist(figsize=(20,15))


```

Listing A.1: Data Processing (the code is presented in simplified form)

```python
#preprocessing
def precond_train(data):

# REMOVING NAN VALUES FROM GROUP FEATURE

# MANIPULATING ANOMALOUS DATA
data['DEPTH_BIT'] = data['DEPTH_BIT'].round(3).replace(-999.25,
np.nan)
data['BLOCK_VEL'] = data['BLOCK_VEL'].round(3).replace(-999.25,
np.nan)
data['BLOCKPOS'] = data['BLOCKPOS'].round(3).replace(-999.25, np.
nan)
data['ROP'] = data['ROP'].round(3).replace(-999.25, np.nan)
data['HKLD'] = data['HKLD'].round(3).replace(-999.25, np.nan)
data['WOB'] = data['WOB'].round(3).replace(-999.25, np.nan)
data['TORQ'] = data['TORQ'].round(3).replace(-999.25, np.nan)
data['DH_ECD'] = data['DH_ECD'].round(3).replace(-999.25, np.nan)
data['PUMP_PRESS'] = data['PUMP_PRESS'].round(3).replace(-999.25,
 np.nan)
data['RPM_SURF'] = data['RPM_SURF'].round(3).replace(-999.25, np.
nan)
data['MUD_FLOW_IN'] = data['MUD_FLOW_IN'].round(3).replace
(-999.25, np.nan)
data['PUMP_STROKE_COUNT'] = data['PUMP_STROKE_COUNT'].round(3).
replace(-999.25, np.nan)

return data

df = precond_train(df)


# Check duplicates:
print('no. of duplicates DB', df['DEPTH_BIT'].duplicated().sum(
axis=0))
print('no. of duplicates HKLD', df['HKLD'].duplicated().sum(axis
=0))
print('no. of duplicates BLOCK_VEL', df['BLOCK_VEL'].duplicated()
.sum(axis=0))
print('no. of duplicates BLOCK_POS', df['BLOCKPOS'].duplicated().
sum(axis=0))
print('no. of duplicates PUMP_P', df['PUMP_PRESS'].duplicated().
sum(axis=0))
print('no. of duplicates MUD_FLOW_IN', df['MUD_FLOW_IN'].
duplicated().sum(axis=0))

print('no. of duplicates RPM', df['RPM_SURF'].duplicated().sum(
axis=0))
print('no. of duplicates ECD', df['DH_ECD'].duplicated().sum(axis
=0))
print('no. of duplicates WOB', df['WOB'].duplicated().sum(axis=0)
)
```

```
  print('no. of duplicates ROP', df['ROP'].duplicated().sum(axis=0))
  print('no. of duplicates PUMP_P', df['PUMP_PRESS'].duplicated().sum(axis=0))
  print('no. of duplicates ROTATING_FRICTION', df['ROTATING_FRICTION'].duplicated().sum(axis=0))
  print('no. of duplicates MUD_FLOW_IN', df['MUD_FLOW_IN'].duplicated().sum(axis=0))
  print('no. of duplicates Pump strokes', df['PUMP_STROKE_COUNT'].duplicated().sum(axis=0))

  #MISSING VALUES
  df=df[['TIME','DEPTH_BIT','BLOCK_VEL','BLOCKPOS','ROP','HKLD','WOB','TORQ','DH_ECD','PUMP_PRESS','RPM_SURF','MUD_FLOW_IN','PUMP_STROKE_COUNT']]
  df.head()


```

Listing A.2: Data Processing (the code is presented in simplified form)

```
  #RAW DATA PLOT
  fig = plt.figure(figsize=(25, 10))
  grid = plt.GridSpec(4, 4, hspace=0.9)


 RPM_SURF= fig.add_subplot(grid[0, 0])
 BLOCK_VEL= fig.add_subplot(grid[1, 0])
 BLOCKPOS = fig.add_subplot(grid[2, 0])

 ROP=fig.add_subplot(grid[0, 1])
 HKLD = fig.add_subplot(grid[1, 1])
 WOB = fig.add_subplot(grid[2, 1])

 TORQ = fig.add_subplot(grid[0, 2])
 DH_ECD = fig.add_subplot(grid[1, 2])
 PUMP_PRESS=fig.add_subplot(grid[2, 2])

 MUD_FLOW_IN=fig.add_subplot(grid[0, 3])
 PUMP_STROKE_COUNT=fig.add_subplot(grid[1, 3])

 BLOCK_VEL.scatter(df['DEPTH_BIT'],df['BLOCK_VEL'], s=5)
 BLOCK_VEL.set(xlabel="DEPTH_BIT", ylabel="BLOCK_VEL")

 BLOCKPOS.scatter(df['DEPTH_BIT'],df['BLOCKPOS'], s=5)
 BLOCKPOS.set(xlabel="DEPTH_BIT", ylabel="BLOCKPOS")

 ROP.scatter(df['DEPTH_BIT'],df['ROP'], s=5)
 ROP.set(xlabel="DEPTH_BIT", ylabel="ROP")

 HKLD.scatter(df['DEPTH_BIT'],df['HKLD'], s=5)
 HKLD.set(xlabel="DEPTH_BIT", ylabel="HKLD")
```

```python

    WOB.scatter(df['DEPTH_BIT'],df['WOB'], s=5)
    WOB.set(xlabel="DEPTH_BIT", ylabel="WOB")

    TORQ.scatter(df['DEPTH_BIT'],df['TORQ'], s=5)
    TORQ.set(xlabel="DEPTH_BIT", ylabel="TORQ")

    DH_ECD.scatter(df['DEPTH_BIT'],df['DH_ECD'], s=5)
    DH_ECD.set(xlabel="DEPTH_BIT", ylabel="DH_ECD")

    PUMP_PRESS.scatter(df['DEPTH_BIT'],df['PUMP_PRESS'], s=5)
    PUMP_PRESS.set(xlabel="DEPTH_BIT", ylabel="PUMP_PRESS")

    RPM_SURF.scatter(df['DEPTH_BIT'],df['RPM_SURF'], s=5)
    RPM_SURF.set(xlabel="DEPTH_BIT", ylabel="RPM_SURF")

    MUD_FLOW_IN.scatter(df['DEPTH_BIT'],df['MUD_FLOW_IN'], s=5 )
    MUD_FLOW_IN.set(xlabel="DEPTH_BIT", ylabel="MUD_FLOW_IN")


    PUMP_STROKE_COUNT.scatter(df['DEPTH_BIT'],df['PUMP_STROKE_COUNT'
   ], s=5 )
    PUMP_STROKE_COUNT.set(xlabel="DEPTH_BIT", ylabel="
  PUMP_STROKE_COUNT")

     plt.show()

    #DATA SELECTION
    df = df.iloc[177587:,:]

    figure = plt.figure
    ax = plt.gca()
    ax.scatter(df['TIME_sec'], df['HKLD'])
    ax.set_xlabel("TIME_sec")
    ax.set_ylabel("HKLD")
    ax.set_title("{} vs {}".format("TIME_sec", "HKLD"))
    #ax.set_xlim(1225, 1300)
    plt.legend()
    plt.show()


    # Function that calculates the percentage of missing values
    def calc_percent_NAs(df):
        nans = pd.DataFrame(df.isnull().sum().sort_values(ascending=
   False)/len(df), columns=['percent'])
        idx = nans['percent'] > 0
        return nans[idx]
    # Let's use above function to look at top ten columns with NaNs
    calc_percent_NAs(df)

```

```
```

Listing A.3: Data Processing (the code is presented in simplified form)

```
#RESAMPLING
df1 = df.set_index('TIME').resample('0.5S')['DEPTH_BIT'].mean()
df2 = df.set_index('TIME').resample('0.5S')['BLOCK_VEL'].mean()
df3 = df.set_index('TIME').resample('0.5S')['BLOCKPOS'].mean()
df4 = df.set_index('TIME').resample('0.5S')['ROP'].mean()
df5 = df.set_index('TIME').resample('0.5S')['HKLD'].mean()
df6 = df.set_index('TIME').resample('0.5S')['WOB'].mean()
df7 = df.set_index('TIME').resample('0.5S')['TORQ'].mean()
df8 = df.set_index('TIME').resample('0.5S')['DH_ECD'].mean()
df9 = df.set_index('TIME').resample('0.5S')['PUMP_PRESS'].mean()
df10 = df.set_index('TIME').resample('0.5S')['RPM_SURF'].mean()
df11 = df.set_index('TIME').resample('0.5S')['MUD_FLOW_IN'].mean
()
df12 = df.set_index('TIME').resample('0.5S')['PUMP_STROKE_COUNT'
].mean()

df1=pd.DataFrame(df1)
df2=pd.DataFrame(df2)
df3=pd.DataFrame(df3)
df4=pd.DataFrame(df4)
df5=pd.DataFrame(df5)
df6=pd.DataFrame(df6)
df7=pd.DataFrame(df7)
df8=pd.DataFrame(df8)
df9=pd.DataFrame(df9)
df10=pd.DataFrame(df10)
df11=pd.DataFrame(df11)
df12=pd.DataFrame(df12)
df = pd.concat([df1, df2, df3, df4, df5, df6, df7, df8, df9, df10
, df11, df12],axis=1)
df=df. reset_index()


YEAR_base=df['TIME'].dt.year
month=df['TIME'].dt.month
day=df['TIME'].dt.day


hour_base=df['TIME'].dt.hour[1]
minute_base=df['TIME'].dt.minute[1]
seconds_base=df['TIME'].dt.second[1]
microseconds_base=df['TIME'].dt.microsecond[1]/1000000

hour=df['TIME'].dt.hour
minute=df['TIME'].dt.minute
seconds=df['TIME'].dt.second
microseconds=df['TIME'].dt.microsecond/1000000

```

```
    hour=hour*60*60-(hour_base*60*60)
    minute=minute*60-(minute_base*60)
    total_seconds=+hour+minute+seconds+microseconds-seconds_base-microseconds_base
    print(total_seconds)
    df.insert(0, 'TIME_sec', total_seconds)

    x_col = "TIME_sec"
y_columns = ["DEPTH_BIT","BLOCK_VEL","BLOCKPOS","ROP","HKLD","WOB","TORQ","DH_ECD","PUMP_PRESS","RPM_SURF","MUD_FLOW_IN","PUMP_STROKE_COUNT"]


    for y_col in y_columns:
        figure = plt.figure
        ax = plt.gca()
        ax.scatter(df[x_col], df[y_col])
        ax.set_xlabel(x_col)
        ax.set_ylabel(y_col)
        ax.set_title("{} vs {}".format(x_col, y_col))
        ax.set_xlim(0, 200)
        plt.legend()
        plt.show()


```

Listing A.4: Data Processing (the code is presented in simplified form)

```
    #INTERPOLATION
    df['DEPTH_BIT'] = df['DEPTH_BIT'].interpolate()
    df['BLOCK_VEL'] = df['BLOCK_VEL'].interpolate()
    df['BLOCKPOS'] = df['BLOCKPOS'].interpolate()
    df['ROP'] = df['ROP'].interpolate()
    df['HKLD'] = df['HKLD'].interpolate()
    df['WOB'] = df['WOB'].interpolate()
    df['TORQ'] = df['TORQ'].interpolate()
    df['DH_ECD'] = df['DH_ECD'].interpolate()
    df['PUMP_PRESS'] = df['PUMP_PRESS'].interpolate()
    df['RPM_SURF'] = df['RPM_SURF'].interpolate()
    df['MUD_FLOW_IN'] = df['MUD_FLOW_IN'].interpolate()
    df['PUMP_STROKE_COUNT'] = df['PUMP_STROKE_COUNT'].interpolate()

```

Listing A.5: Data Processing (the code is presented in simplified form)

```
    #Noise reduction
    df_name=['TIME_sec','DEPTH_BIT','BLOCK_VEL','BLOCKPOS','ROP','HKLD','WOB','TORQ','DH_ECD','PUMP_PRESS','RPM_SURF','MUD_FLOW_IN','PUMP_STROKE_COUNT']
    x=10
    for i in df_name:
        df[i] = df[i].rolling(window = x, center=True).mean()
    df.dropna(inplace = True)

    #save dataframe
    df_time_print = df
    df_time_print.to_csv(r'DATA_CLEANED_TIME.csv', index = False,
header=True)


```

Listing A.6: Data Processing (the code is presented in simplified form)

A.3 In-Slips detection

```
    # Preprocessed data
    df = pd.read_csv('DATA_CLEANED_TIME.csv',sep=',',parse_dates=['
TIME'])

    #Traditional method: Interquartile Range (IQR)
    # Calculate IQR for the 1st principal component (pc1)
    q1_pc1, q3_pc1 = df['HKLD'].quantile([0.25, 0.75])
    iqr_pc1 = q3_pc1 - q1_pc1
    # Calculate upper and lower bounds for outlier for pc1
    lower_pc1 = q1_pc1 - (1.5*iqr_pc1)
    upper_pc1 = q3_pc1 + (1.5*iqr_pc1)
    # Filter out the outliers from the pc1
    df['anomaly_HKLD'] = ((df['HKLD']>upper_pc1) | (df['HKLD']<
lower_pc1)).astype('int')


    # Let's plot the outliers from pc1 on top of the sensor_11 and
see where they occured in the time series
    a = df[df['anomaly_HKLD'] == 1] #anomaly
    _ = plt.figure(figsize=(18,6))
    _ = plt.plot(df['HKLD'], color='blue', label='Out-of-slips')
    _ = plt.plot(a['HKLD'], linestyle='none', marker='X', color='red'
, markersize=12, label='In-slips')
    _ = plt.xlabel('Date and Time')
    _ = plt.ylabel('Sensor Reading')
    _ = plt.title('HKLD Sensor Anomalies')
    _ = plt.legend(loc='best')
    plt.show();

    # Calculate IQR for the 2nd principal component (pc2)
    q1_pc2, q3_pc2 = df['BLOCKPOS'].quantile([0.4, 0.6])
    iqr_pc2 = q3_pc2 - q1_pc2
    # Calculate upper and lower bounds for outlier for pc2
    lower_pc2 = q1_pc2 - (1.2*iqr_pc2)
    upper_pc2 = q3_pc2 + (1.2*iqr_pc2)
    # Filter out the outliers from the pc2
    df['anomaly_BLOCKPOS'] = ((df['BLOCKPOS']>upper_pc2) | (df['BLOCKPOS']<lower_pc2)).astype('int')


    # Let's plot the outliers from pc1 on top of the sensor_11 and see where they occured in the time series
    a = df[df['anomaly_BLOCKPOS'] == 1] #anomaly
    _ = plt.figure(figsize=(18,6))
    _ = plt.plot(df['BLOCKPOS'], color='blue', label='Normal')
    _ = plt.plot(a['BLOCKPOS'], linestyle='none', marker='X', color='red', markersize=12, label='Anomaly')
    _ = plt.xlabel('Date and Time')
    _ = plt.ylabel('Sensor Reading')
    _ = plt.title('BLOCKPOS Sensor Anomalies')
    _ = plt.legend(loc='best')
    plt.show();


```

Listing A.7: Pattern Recognition (the code is presented in simplified form)

```

    #Pattern recognition method
    # Calculate IQR for the 2nd principal component (pc2)
    q1_pc2, q3_pc2 = df['BLOCKPOS'].quantile([0.4, 0.6])
    iqr_pc2 = q3_pc2 - q1_pc2
    # Calculate upper and lower bounds for outlier for pc2
    lower_pc2 = q1_pc2 - (1.2*iqr_pc2)
    upper_pc2 = q3_pc2 + (1.2*iqr_pc2)
    # Filter out the outliers from the pc2
    df['anomaly_BLOCKPOS'] = ((df['BLOCKPOS']>upper_pc2) | (df['BLOCKPOS']<lower_pc2)).astype('int')


    # Let's plot the outliers from pc1 on top of the sensor_11 and see where they occured in the time series
    a = df[df['anomaly_BLOCKPOS'] == 1] #anomaly
    _ = plt.figure(figsize=(18,6))
    _ = plt.plot(df['BLOCKPOS'], color='blue', label='Normal')
    _ = plt.plot(a['BLOCKPOS'], linestyle='none', marker='X', color='red', markersize=12, label='Anomaly')
    _ = plt.xlabel('Date and Time')
    _ = plt.ylabel('Sensor Reading')
    _ = plt.title('BLOCKPOS Sensor Anomalies')
    _ = plt.legend(loc='best')
    plt.show();


```

Listing A.8: Pattern Recognition (the code is presented in simplified form)

```
#PATTERN RECOGNITION
#Changepoint detection with the Binary Segmentation search method
model = "l2"
algo = rpt.Binseg(model=model).fit(points)
my_bkps = algo.predict(n_bkps=16)

# show results
rpt.show.display(points, my_bkps, figsize=(10, 6))
plt.title('Change Point Detection: Binary Segmentation Search Method')
plt.xlabel('Number of HKLD values for CPD')
plt.ylabel('HKLD Sensor Reading')
plt.show()

```

Listing A.9: Pattern Recognition (the code is presented in simplified form)

```
#FILTER IN-SLIPS FROM OUT-OF-SLIPS
ls = [i for i in range(0,my_bkps[0])] + [i for i in range(my_bkps[1],my_bkps[2])] + [i for i in range(my_bkps[3],my_bkps[4])]+[i for i in range(my_bkps[5],my_bkps[6])]+[i for i in range(my_bkps[7],my_bkps[8])]+[i for i in range(my_bkps[9],my_bkps[10])]+[i for i in range(my_bkps[11],my_bkps[12])]+[i for i in range(my_bkps[13],my_bkps[14])]+[i for i in range(my_bkps[15],my_bkps[16])]

df=df[df.index.isin(ls)]

#SAVE IN .CSV
df_print = df
df_print.to_csv(r'OUT_slips.csv', index = False, header=True)

```

Listing A.10: Pattern Recognition (the code is presented in simplified form)

A.4 Unsupervised Machine Learning Implementation

```
#IMPORT DATA
df_classif = pd.read_csv('OUT_slips.csv',sep=',',parse_dates=['TIME'])

```

Listing A.11: Unsupervised ML Clustering (the code is presented in simplified form)

```
#FUZZY C-MEANS
df_analysis = df_classif[(df_classif['TIME_sec'] >= 0) & (df_classif['TIME_sec'] <= 1200)]

petal_df=df_fuzzy[['TIME_sec','BLOCKPOS','BLOCK_VEL','HKLD','RPM_SURF','TORQ','PUMP_PRESS','DH_ECD','MUD_FLOW_IN','PUMP_STROKE_COUNT']]
petal_df= np.array(petal_df)
fcm1 = FCM(n_clusters=7)
fcm1.fit(petal_df)

# outputs
fcm_centers = fcm1.centers
fcm_labels = fcm1.predict(petal_df)

# plot result
f, axes = plt.subplots(1, 2, figsize=(12,5))
axes[0].scatter(petal_df[:,0], petal_df[:,3], alpha=.1, c='g', label ='HKLD behavior after the first connection')
axes[0].legend()

axes[1].scatter(petal_df[:,0], petal_df[:,3], label ='Tripping in', c=fcm_labels, alpha=.1, marker="o")
axes[1].legend()
axes[1].scatter(fcm_centers[:,0], fcm_centers[:,3], label ='Centers of clusters', s=500, c='r',marker="+")
axes[1].legend()
plt.title('Estimated HKD - Clustering')
plt.xlabel ('TIME (s)')
plt.ylabel ('Hookload (Tonne)')
plt.show()

petal_df=pd.DataFrame(petal_df, columns=['TIME_sec','BLOCKPOS','BLOCK_VEL','HKLD','RPM_SURF','TORQ','PUMP_PRESS','DH_ECD','MUD_FLOW_IN','PUMP_STROKE_COUNT'])
petal_df['Identification'] = fcm_labels.tolist()

```

Listing A.12: Unsupervised ML Clustering (the code is presented in simplified form)

```
#K-MEANS
```

```python
    df_kmeans= df_classif[(df_classif['TIME_sec'] >= 0) & (df_classif
   ['TIME_sec'] <= 1200)]

    df_kmeans_narrow=df_kmeans[['TIME_sec','BLOCKPOS','BLOCK_VEL','
  HKLD','RPM_SURF','TORQ','PUMP_PRESS','DH_ECD','MUD_FLOW_IN','
  PUMP_STROKE_COUNT']]
   X = df_kmeans_narrow

   K=8
   # Select random observation as centroids
   Centroids = (X.sample(n=K))
   plt.scatter(X["TIME_sec"],X["HKLD"],c='black')
   plt.scatter(Centroids["TIME_sec"],Centroids["HKLD"],c='red')
   plt.xlabel ('TIME (s)')
   plt.ylabel ('Hookload (Tonne)')
   plt.title('K-means Clustering')
   plt.show()


   diff = 1
   j=0
   while(diff!=0):
        XD=X
        i=1
        for index1,row_c in Centroids.iterrows():
             ED=[]
             for index2,row_d in XD.iterrows():
                  d1=(row_c["TIME_sec"]-row_d["TIME_sec"])**2
                  d2=(row_c["HKLD"]-row_d["HKLD"])**2
                  d=np.sqrt(d1+d2)
                  ED.append(d)
             X[i]=ED
             i=i+1

        C=[]
        for index,row in X.iterrows():
             min_dist=row[1]
             pos=1
             for i in range(K):
                  if row[i+1] < min_dist:
                       min_dist = row[i+1]
                       pos=i+1
             C.append(pos)
        X["Cluster"]=C
        Centroids_new = X.groupby(["Cluster"]).mean()[["HKLD","
  TIME_sec"]]
        if j == 0:
             diff=1
             j=j+1
        else:
             diff = (Centroids_new['HKLD'] - Centroids['HKLD']).sum()
```

```
+ (Centroids_new['TIME_sec'] - Centroids['TIME_sec']).sum()
            print(diff.sum())
        Centroids = X.groupby(["Cluster"]).mean()[["HKLD","TIME_sec"
]]

 color=['black','blue','blue','blue','blue','blue','blue','blue']
 for k in range(K):
        data=X[X["Cluster"]==k+1]
        plt.scatter(data["TIME_sec"],data["HKLD"],c=color[k], label =
'Tripping down')

 plt.scatter(Centroids["TIME_sec"],Centroids["HKLD"],c='red',
label ='Centers of clusters')
 plt.xlabel ('TIME (s)')
 plt.ylabel ('Hookload (Tonne)')
 plt.title('K-means Clustering')
 plt.show()
```

Listing A.13: Unsupervised ML Clustering (the code is presented in simplified form)

```
 #GMM - Gaussian Mixture Modelling
 df_gmm = df_classif[(df_classif['TIME_sec'] >= 0) & (df_classif['
TIME_sec'] <= 1200)]

 df_gmm=df_gmm[['TIME_sec','BLOCKPOS','BLOCK_VEL','HKLD','RPM_SURF
','TORQ', 'PUMP_PRESS','DH_ECD','MUD_FLOW_IN','PUMP_STROKE_COUNT'
]]

 #Standardize the data to normal distribution
 dataset1_standardized = preprocessing.scale(df_gmm)
 dataset1_standardized = pd.DataFrame(dataset1_standardized)
 print(dataset1_standardized)

 # Create the gmm model with the selected number of clusters/
components
 gmm = GaussianMixture(n_components=6)

 # Fit the model to our dataset
 gmm.fit(dataset1_standardized)

 # Predict the labels
 gmm_labels = gmm.predict(dataset1_standardized)

 # Assign the labels back to the workingdf
 df_gmm['GMM'] = gmm_labels

 color=['black','blue','blue','blue','blue','blue']
 K=6
 for k in range(K):
        k=k-1
        data=df_gmm[df_gmm["GMM"]==k+1]
        plt.scatter(data["TIME_sec"],data["HKLD"],c=color[k], label =
    'Tripping down')

        plt.xlabel ('TIME (s)')
        plt.ylabel ('Hookload (Tonne)')
        plt.title('GMM Clustering')
        plt.show()


```

Listing A.14: Unsupervised ML Clustering (the code is presented in simplified form)

```
    #SAVE IN .CSV
    df_print = df_full_kmeans
    df_print.to_csv(r'clustered_data.csv', index = False, header=True
    )

```

Listing A.15: Unsupervised ML Clustering (the code is presented in simplified form)

A.5 Catching the trend

```python
df_label = pd.read_csv('clustered_data.csv',sep=',',parse_dates=['TIME'])
print('shape:', df_label.shape)

#Visualization: period we pick-up string until we tag bottom for every connection.
color=['black']
k=1
for k in range(K):
    k=k
    data=df_label [df_label ["Cluster"]==+1]
    plt.scatter(data["TIME_sec"],data["HKLD"],c=color[k], label = 'Tripping down')

plt.xlabel ('TIME (s)')
plt.ylabel ('Hookload (Tonne)')
plt.title('K-means Clustering')
plt.show()

```

Listing A.16: Free-rotating HKLD Region Detection (the code is presented in simplified form)

```python
#Create a dataframe for every connection (8 connections in total)
data
data['Indexing']=data.index
data = data.reset_index()
data = data.drop(['index'], axis=1)

l=0
for k, g in groupby(enumerate(data['Indexing']), lambda i_x: i_x[0] - i_x[1]):
    mapping=list(map(itemgetter(1), g))
    print(mapping)

l=0
for k, g in groupby(enumerate(data['Indexing']), lambda i_x: i_x[0] - i_x[1]):
    mapping[l]=list(map(itemgetter(1), g))
    l=l+1
    print(mapping[l])


df0=data.loc[data['Indexing'].isin(mapping[0])]
df0 = df0.drop(['Indexing'], axis=1)
df1=data.loc[data['Indexing'].isin(mapping[1])]
df1 = df1.drop(['Indexing'], axis=1)
df2=data.loc[data['Indexing'].isin(mapping[2])]
df2 = df2.drop(['Indexing'], axis=1)
df3=data.loc[data['Indexing'].isin(mapping[3])]
```

```
    df3 = df3.drop(['Indexing'], axis=1)
    df4=data.loc[data['Indexing'].isin(mapping[4])]
    df4 = df4.drop(['Indexing'], axis=1)
    df5=data.loc[data['Indexing'].isin(mapping[5])]
    df5 = df5.drop(['Indexing'], axis=1)
    df6=data.loc[data['Indexing'].isin(mapping[6])]
    df6 = df6.drop(['Indexing'], axis=1)
    df7=data.loc[data['Indexing'].isin(mapping[7])]
    df7 = df7.drop(['Indexing'], axis=1)
    df8=data.loc[data['Indexing'].isin(mapping[8])]
    df8 = df8.drop(['Indexing'], axis=1)



```

Listing A.17: Free-rotating HKLD Region Detection (the code is presented in simplified form)

```
    #Pre-set statistical analysis for one connection.
    n=0
    variance_pp = np.empty(shape=(50000, 1), dtype=float)
    for windowpp in df0['PUMP_PRESS'].rolling(window=5):
        variance_pp[n]=windowpp.var()
        print(variance_pp[n])
        n=n+1

    n=0
    variance_flow = np.empty(shape=(50000, 1), dtype=float)
    for windowflow in df0['MUD_FLOW_IN'].rolling(window=5):
        variance_flow[n]=windowflow.var()
        print(variance_flow[n])
        n=n+1

    n=0
    variance_rpm = np.empty(shape=(50000, 1), dtype=float)
    mean_rpm= np.empty(shape=(50000, 1), dtype=float)

    for windowrpm in df0['RPM_SURF'].rolling(window=5):
        mean_rpm[n]=windowrpm.mean()
        variance_rpm[n]=windowrpm.var()
        print(variance_rpm[n])
        print(mean_rpm[n])
        n=n+1

    df0_ana=pd.DataFrame()
    for x in range(0, n):
        print("We're on %d" % (x))
        if (mean_rpm[x]>90 and variance_rpm[x]<80 and (variance_flow
  [n]<80) and (variance_pp[x]<0.5)):
            print("STOP")
            df0_ana[x]=df0.iloc[x]

    df0_ana=df0_ana.T
    print("\n----------- Calculate
Mean------------\n")
    print(df0_ana["HKLD"].mean())

    print("\n----------- Calculate Median----------\n")
    print(df0_ana["HKLD"].median())

    print("\n----------- Calculate Variance -----------\n")
    print(df0_ana["HKLD"].var())

```

Listing A.18: Free-rotating HKLD Region Detection (the code is presented in simplified form)

```
#Visualization of the free-rotating HKLD region
names=df0.columns
for name in names:
    ax = df0.plot(x='TIME_sec', y=[name], kind='scatter', c='r',
label='Small')
    df0_ana.plot(x='TIME_sec', y=[name], kind='scatter', ax=ax, c
='g', label='Free rotation')
    _ = plt.title(name)
    _ = plt.xlabel('Time')
    _ = plt.ylabel('Sensor Reading')
    plt.show()


```

Listing A.19: Free-rotating HKLD Region Detection (the code is presented in simplified form)

```
#HKLD Mean value for every Free-rotating region. Furthermore,
dataframe is created.
data = {'Mean_HKLD': [df0_ana["HKLD"].mean(), df1_ana["HKLD"].
mean(), df2_ana["HKLD"].mean(), df3_ana["HKLD"].mean(), df4_ana["
HKLD"].mean(),df5_ana["HKLD"].mean(),df6_ana["HKLD"].mean(),
df7_ana["HKLD"].mean(),df8_ana["HKLD"].mean()],
    'TIME':[df0_ana['TIME_sec'].mean(),df1_ana['TIME_sec'].mean(),
df2_ana['TIME_sec'].mean(),df3_ana['TIME_sec'].mean(),df4_ana['
TIME_sec'].mean(),df5_ana['TIME_sec'].mean(),df6_ana['TIME_sec'].
mean(),df7_ana['TIME_sec'].mean(),df8_ana['TIME_sec'].mean()]
    }

df = pd.DataFrame(data)

plt.scatter(df["TIME"],df["Mean_HKLD"],c='black')
plt.xlabel('Time')
plt.ylabel('HKLD')
plt.show()

```

Listing A.20: Trend Estimation (the code is presented in simplified form)

```
#Linear Regression - FIND TREND
#Change of the Free-rotating HKLD Trend for the Drilling Phase I
x = np.array(df["TIME"]).reshape((-1, 1))
y = np.array(df["Mean_HKLD"])
model = LinearRegression()
model.fit(x, y)
r_sq = model.score(x, y)
print(f"coefficient of determination: {r_sq}")
print(f"intercept: {model.intercept_}")
print(f"slope: {model.coef_}")

df_04RL=df.iloc[:4,:]
x04 = np.array(df_04RL["TIME"]).reshape((-1, 1))
y04 = np.array(df_04RL["Mean_HKLD"])
regression_model = LinearRegression()
regression_model.fit(x04, y04)
y_predicted04 = regression_model.predict(x04)
rmse = mean_squared_error(y04, y_predicted04)
r2 = r2_score(y04, y_predicted04)
plt.scatter(x04, y04, s=10)
plt.xlabel('HOOKLOAD')
plt.ylabel('TIME (seconds)')
plt.plot(x04, y_predicted04, color='r')

plt.scatter(x02, y02, s=80)
plt.scatter(x03, y03, s=80)
plt.scatter(x04, y04, s=80)
plt.ylabel('HOOKLOAD (tonn)')
plt.xlabel('TIME (seconds)')
plt.plot(x04, y_predicted04, color='y')
plt.plot(x02, y_predicted02, color='r')
plt.plot(x03, y_predicted03, color='b')
plt.title('Trend - Drilling Phase I')
plt.show()
```

Listing A.21: Trend Estimation (the code is presented in simplified form)