FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Computational Engineering	Spring semester, 2021 Open / ~~Confidential~~
Author: Winusan Wijayaseelan	(signature of author)
Faculty Supervisors: Kjell Kåre Fjelde and Mesfin Belayneh External Supervisors: Dalila Gomes and Tim Robinson	
Title of master's thesis: Use of Machine Learning Algorithms for Stuck Pipe Prediction	
Credits: 30	
Keywords: Machine Learning, Stuck Pipe, LSTM, Differential Sticking, ANN, Hook load, HKL, T&D, ML.	Number of pages………68… Stavanger

Abstract

The oil and gas (O&G) industry have over the course of the years experienced a tremendous advancement on its technological front. The swift progression has provided the industry with the necessary tools to reduce cost and non-productive time (NPT), while being able to increase the operational efficiency. However, stuck pipe is still one the leading contributor to downtime. New and improved solutions to prevent pipe sticking is therefore continuously sought out to further optimize ongoing drilling operation. Because of this, utilization of predictive machine learning algorithms integrated in real-time software has become a topic of interest.

This thesis project investigates the prospects regarding further development of the established Artificial Neural Network (ANN) models by Exebenus, a digitalization company in the O&G industry. Their models for differential sticking, wellbore geometry, and hole cleaning have already proven to be of great assistance in vertical and deviated wells. Therefore, opportunities to expand their differential sticking model to horizontal wells have been explored throughout this paper. A univariate time-series model was developed, tested, and evaluated to predict the hookload trend ahead of time. In addition, expected changes due to increased inclination, borehole friction and resistive force were considered using a discretized torque and drag model. With a mean absolute error of 6.73 tonnes, the output of the machine learning model provided in this thesis proved little to no systematic offsets and were capable of predicting the hookload, making it applicable for horizontal well sections.

Acknowledgement

I would like to express my sincere gratitude towards Kjell Kåre Fjelde and Mesfin Belayneh, your supervision, guidance and advice throughout this research project were of more help than you could imagine, and I would not have been able to completed this task without your help.

I would also like to thank Exebenus for providing me with such an interesting thesis topic, which allowed me to use my knowledge within the field of Computational Engineering and Petroleum Technology. A special thanks to Dalila Gomes and Tim Robinson, for providing me with assistance even throughout the difficult times which the pandemic has brought upon us.

To my friends, who spent most of the days alongside with me as this thesis was written. Thank you for the lively conversations, interesting discussion, and motivational speeches when things looked bliss. Your presence made life interesting both on and off campus, and my days as a student would not have been the same without you.

Lastly, I would like to thank my family for their wise counsel and sympathetic ear. The never-ending support and faith you have in me is something that I'll always treasure.

Table of Contents

List of Figures

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
AAPG	Association of Petroleum Geologist
API	Application Programming Interface
BHA	Bottom Hole Assembly
BOP	Blowout Preventer
CT	Coiled Tubing
DS	Differential Sticking
DT	Decision Tree
ECD	Equivalent Circulating Density
E&P	Energy and Petroleum
GB	Gradient Boosting
HC	Hole Cleaning
HKL	Hookload
HWDP	Heavy Weight Drill Pipe
ID	Inner Diameter
KNN	K-Nearest Neighbor
LGS	Low-Gravity Solids
LOOCV	Leave-One-Out Cross-Validation
LPOCV	Leave-P-Out Cross-Validation
LR	Logistic Regression
LSTM	Long Short-Term Memory
ML	Machine Learning
MLR	Multi-Linear Regression
MAE	Mean Absolute Error
MWD	Measurement-While-Drilling
NCS	North Continental Shelf
NPT	Non-Productive Time

OD	Outer Diameter
O&G	Oil and Gas
POOH	Pull Out of Hole
ReLU	Rectified Linear Activation Function
RF	Random Forest
RIH	Run in Hole
RNN	Recurrent Neural Network
ROP	Rate of Penetration
SPP	Standpipe Pressure
SVM	Support Vector Machine
TanH	Hyperbolic Tangent Activation Function
T&D	Torque and Drag
WBM	Water-Based Mud
WG	Wellbore Geometry
WL	Wireline
WOB	Weight on Bit
WOH	Weight on Hook
WOW	Waiting on Weather

Nomenclature Drilling

ρ	Density
β	Buoyancy factor
μ	Coefficient of friction
μ	Fluid Viscosity
$\Delta\alpha$	Wellbore Azimuth
$\Delta\theta$	Wellbore Inclination
ΔP	Differential Pressure
A	Cross sectional area
A_c	Drill collar contact area
F	Force
F_{dl}	Deadline Tension
F_N	Normal Force
g	Gravitational force
L	Length of drillstring component
P_{FF}	Formation pore pressure
P_M	Pressure of mud column
R	Radius
W	Buoyed weight
W	Mass
u	Fluid Velocity

Nomenclature Machine Learning

μ	Momentum
A	Hidden layer state
C	Cell State
f^t	Forget Gate
h	Hidden layer
I^t	Current Input Data
i^t	Input Gate Sigmoid
j	Action Potential
k	Output layer
m	Output Lay
O^t	Output Gate
H	Action Potential Vector
I	Input Vector
B	Bias Vector
F	Activation Function
O	Output Vector
W	Weight Vector
δk	Error Gradient
t	Predicted target output
w	Weight of Node
$\hat{y}_I$	Predicted Output Value
y_i	True Output V

1 Introduction

1.1 Background

Stuck pipe is recognized as one of the largest contributors to nonproductive time (NPT) and is a major expenditure in the oil and gas industry. Estimates indicate that stuck pipe incidences costs the petroleum sector more than $250 000 000 annually (Bradley et al., 1991). In addition, problems related to pipe sticking can account for almost half of the overall well cost, making stuck pipe one of the most expensive problems that can take place during a drilling operation (Mitchell, 2011). Because of this, early detection of possible stuck pipe occurrences is essential to prevent a decline in drilling efficiency and unnecessary large well costs.

Artificial intelligence (AI) and machine learning (ML) and have over the course of the past years grown in popularity across several industries. The energy and petroleum (E&P) sector are no exception, as they are currently in the early stages of applying AI and ML models for various applications (Pathak et al., 2021). The rise in popularity of ML applications in the E&P sector is due to advancement in sensor technology and high-performance computing services that allows procurement of big data and storage in different fields of study (Belyadi and Haghighat, 2021). Big data is a term that is often used in the E&P sector and refers to quantities of data that are too large to be gathered, stored, and analyzed through common tools and processing methods. A brief search on publications in the oil and gas (O&G) industry with the American Association of Petroleum Geologist (AAPG) or Society of Petroleum Engineer's OnePetro can confirm that the number of articles regarding this domain has increased rapidly throughout the past years. As more firms starts to comprehend the added value of incorporating ML into daily operations, the technological advancement in this area is expected to improve.

The biggest concern in terms of stuck pipe situations are the fact that there exists a multitude of combinations which can cause the drillstring to go stuck. As of today, the most approved approach to prevent sticking is the use of analytical models which approximates the conditions of the wellbore. The modeled data can be considered as a guideline, which real-time data at the rig-site is continuously monitored against. The rig-crew then needs to take action if the real-time drilling data starts to deviate from the analytical models. These deviations can in many cases be difficult to detect, as even minor changes can trigger a stuck pipe event. Thus, conventional methods are heavily reliant on an experienced rig-crew in order to make quick and informed decisions to minimize the risk of getting stuck. Machine learning models on the other hand are trained using real-time data to predict future outcomes. By having a machine learning model integrated in the rig software can therefore offer the rig crew with valuable information ahead of time, providing them with room to alleviate the problem before it has even occurred.

1.2 Problem Description

This thesis is written in cooperation with Exebenus, a digitalization company in the O&G sector that have been working on ML models for stuck pipe prediction in vertical wells. In correspondence with existing research and studies on this topic, the company have chosen to develop their models based on the main mechanisms that trigger pipe sticking. Thus, three ML-based models have been developed for this purpose, namely:

- Differential Sticking (DS) model
- Wellbore Geometry (WG) model
- Hole Cleaning (HC) model

where the DS and WG models base their predictions on available Hookload (HKL) data, while their HC model uses flow rate, standpipe pressure (SPP), Rate of Penetration (ROP), and mud weight to calculate the estimated downhole equivalent circulating density (ECD) for forecasting purposes (Meor Hashim et al., 2021b). Although all three models are commercially viable, their area of application varies. The DS models are available for situations related to drilling and tripping, the WG model is offered for tripping scenarios only, while the HC model works for all situations defined as drilling and non-drilling (Meor Hashim et al., 2021b). Furthermore, the DS and HC models are capable of predicting 2 stands ahead during drilling, and all three models are capable of predicting up to 10 stands ahead for a number of tripping operations (Meor Hashim et al., 2021b).

As of today, Exebenus have only implemented their models for vertical and slightly deviated wells. For horizontal wells, which often are defined as wells with inclination above 80 degrees, there exists unanswered question whether the company's ML models are capable of handling lateral well-sections. The concerns regarding this topic are related to known physical changes which occurs in the drilling parameters that the company uses, as the wellbore trajectory transitions from vertical to horizontal.

1.3 Thesis Objective and Approach:

The main objective of this thesis is to build, train and test a machine learning model based on the information provided by Exebenus, to verify whether their models are scalable to horizontal wells or not. To achieve the primary objective, this thesis needs to cover both data science and petroleum related subjects. Therefore, this paper has been written in such a way that newcomers to either fields can acquire a general familiarity and basic understanding of each topic. Hence, the first two chapters will cover an overview of the drilling process and the range of mechanisms that can trigger various stuck pipe situations. The following chapter will cover the topic of machine learning, which includes typical workflow, model types, and necessary procedures to build and access a machine learning model. From chapter four and onwards, details surrounding expected changes related to relevant physics, model architecture and specifics directed towards the experimental set up will be covered.

1.4 Limitations of the Project

Out of the three models, the scope of this thesis will primarily focus on model development, testing and physics surrounding the DS model. However, the general topics presented in this thesis will cover the broader aspects in terms of stuck pipe for all three models. In addition, the differential sticking model developed by Exebenus is comprised of two separate ML models, where one models predicts the output for tripping operations, while the other model predicts the output for drilling operations. This project has been limited to developing a single model, which will attempt to make predictions for both operation which has been separated by the company.

2 Overview of the Drilling Process

Construction of a well requires expertise of people from various fields. In order to secure proper integrity of a well and minimize risks associated with drilling, it is essential to follow the recommended guidelines and operational standards where several, detailed considerations are accounted for. If the execution is done correctly, a properly designed and constructed well should be able to withstand any abnormal condition and unforeseen event (Aadnoy, 2011). Although the operational practice and design may vary from each company and its respective fields, the core activities, as well as their systems, will remain the same. To attain a better understanding of how a drilling operation is performed, rig systems, the drilling process, and important surface parameters will be discussed in the following subchapters.

2.1 Rig System

As of today, hydrocarbon extraction from offshore fields are performed using a variety floating and bottom supported production systems. Application areas for each individual marine rig, offshore platform, or offshore drilling rig depends on factors such as water depth and field size. However, they are most commonly divided into two main categories, namely fixed/bottom supported and floating vessels. Whether it's a floating or fixed vessel, drilling is ensued using method called rotary drilling (Assadi et al., 2019). Essentially, rotary drilling is a technique where rock cutting equipment is mounted to the bottom of a drillstring. The drillstring is hollow steel tubular that provides constant inflow of fluid throughout the rock-cutting tool, called a drillbit, so sediments that loosens when the bit penetrates the formation is able to be transported up to the surface. Offshore production systems that supports rotary drilling is composed of several set of major facilities, and are commonly divided into six main categories (Mitchell and Miska, 2011):

- Power system
- Hoisting system
- Circulating system
- Rotary system
- Well control system
- Well monitoring system

All six are considered essential during drilling, as each system is highly dependent of one another. Special marine equipment is also compulsory in an offshore environment, and the main functionalities for each of these categories will be discussed below, with the exception of the well monitoring system, which will be discussed separately in chapter 2.3.

2.1.1 The Power System

Drilling rigs are relatively large vessels and does naturally require great amounts of energy to be able to function properly. In addition, they are often remotely located to areas with lack of available power supply. Because of this, drilling rigs must have a source of power to fuel the systems on-site. In recent years, electrification of fixed offshore installations from land by means of underwater cables have become a possibility on the north continental shelf (NCS). A prime example of this is the Johan Sverdrup field. However, electrification using landlines is still not a standard procedure in other parts of the world. This is because a large majority of platforms are moving vessels, making them dependent of on-site power sources as they often need to relocate. Furthermore, modern batteries do not have the capacity to power platforms by themselves, and fixed vessels are also required to have back-up generators installed on NCS rig-sites in case the powerlines should fail. Therefore, the most frequently used power source on offshore rigs are diesel-electric systems (Mitchell and Miska, 2011).

A diesel-electric system, or prime mover, works as an internal combustion engine that provides power to drawworks, rotary table, mud pumps and hydraulic equipment present on the rig-site (Assadi et al., 2019). The total number of prime movers necessary to supply sufficient amounts of power to a rig depends solely on the rig-size and its total capacity. The main reason that electrical power sources are used is because their transmission systems transfer energy smoother than mechanical transmission systems (Mitchell and Miska, 2011). This allows the driller to apply power in a controlled manner to the desired rig equipment, hence minimizing shock and vibration problems.

2.1.2 Hoisting System

The hoisting systems is most easily described as a large pulley system which is specifically designed to lift varying types of long, hollow steel tubulars in or out the well, as well as other equipment if an operational situation should require so. The abovementioned hollow tubulars are most commonly known as drillstrings or casings, and their applications will be discussed later in chapter 2.2.

Figure 2.1 shows a hoisting system where the main components, with the exception of the derrick, are displayed, but also supplementary weight-supporting equipment such as the hook, elevators and deadline.

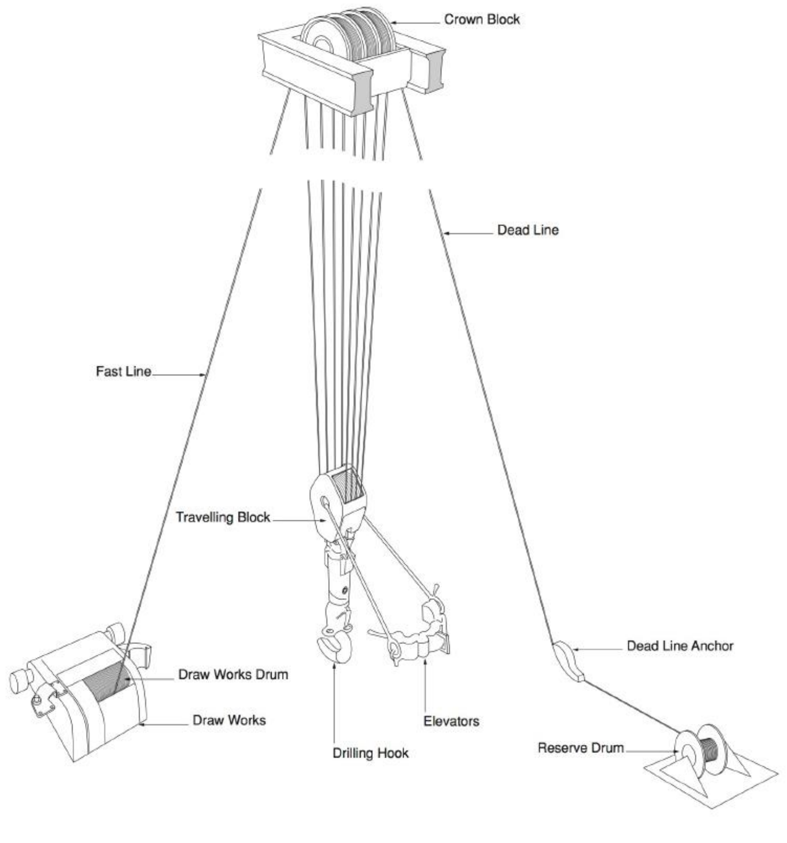

Figure 2.1: *The Hoisting System (Assadi et al., 2019)*

As mentioned in the previous subchapter, the draw-works are supplied with energy from the power system. This power is used to lift pipes or other equipment into position by reeling the drilling line, a spooled, steel-wired line which can be found around a drum residing in the draw-works. The opposite also holds true, as the draw works can be spooled out again when an item needs to be lowered into the well (Assadi et al., 2019). The hoisting system is expected to carry heavy loads during a drilling operation. Thus, the draw-works need to be equipped with a strong, reliable, mechanical braking system to decelerate the applied loads as they are hoisted or lowered into the well. Furthermore, the main braking system are often accommodated with eddy-current brakes on electric rigs, which utilizes two opposing electromagnetic fields to assist the mechanical brakes in the deceleration process (Mitchell and Miska, 2011).

Block and tackle are a collective term for equipment on the hoisting system which includes the travelling block, drilling line and the crown block. These three components are considered as the key connection between the draw works and the loads that will be raised or lowered into the well (Mitchell and Miska, 2011). The purpose of the block and tackle is to allow easier management of large loads by adding a mechanical advantage. This is done by distributing the load exerted on the hoisting system onto an equal number of lines that are threaded through the travelling and crown block.

Lastly, when describing the hoisting system and its purpose, it is essential to mention the derrick. The derrick resembles a tall mast, which the hoisting system is connected to. The extra height provided by the derrick is necessary for the hoisting system to be able to elevate and lower pipes into the well. During a drilling operation however, efficiency is key. The most commonly hoisted steel tubular, the drillpipe, which connects the rig to the drilling equipment used in the well, comes in sectioned lengths called joints and have a rough length of 30ft (Schlumberger, 2021d). Thus, inserting a single joint at time in the hole becomes counterintuitive. The derrick is therefore designed to be able to accommodate longer sections of joints, called stands. A stand is typically 3 unit-lengths of joints that are screwed together (Schlumberger, 2021g). This slight modification of the derrick increases the efficiency of the hoisting system, as longer lengths of pipe can be inserted or removed from the well in a shorter time interval, making the derrick an important part of a drilling operation (Mitchell and Miska, 2011)

2.1.3 Circulating System

The purpose of a fluid circulation system is to supply sufficient amounts of hydraulic power to the drilling fluid or mud, so that it is able to flow efficiently down through the drillstring and back up to the surface through the annular space between the drillstring and wellbore. When the fluid is travelling back up, the power provided by the circulation system should also be capable of transporting drilled cuttings during a regular drilling operation (Bourgoyne et al., 1986).

When referring to a rig circulation system, the principal set of equipment mentioned are mud pumps, mud pits, mud-mixing equipment, and solids-removal equipment. Mud pumps in particular are produced to fit requirements related to flow rate, horsepower and pressure output (Mitchell and Miska, 2011). In terms of flow rate of the pumps, they must pass a minimum threshold to efficiently clean the hole. However, this is rarely a limiting factor for most drilling operations, with the exception of the two most upper hole-sections which are relatively large in diameter. This also holds true for maximum horsepower provided by the pumps. Pump pressure on the other hand is required to have a high output, because circulation of drilling fluids in horizontally long wells, together with appropriate hole cleaning below the drillbit, is heavily dependent of it. Without a properly working circulation system, there is a high probability of encountering various complications throughout the drilling process.

2.1.4 Rotating System

A rotary system is needed on offshore rigs to provide torque, or more commonly known as rotation, to the drillbit. Thus, any component that provides, or are connected to a tool that contributes to rotation, is part of this system. Originally, rotary tables were used as the main driver in all rig systems. Nowadays, all modern rigs have replaced rotary tables with topdrives. The topdrive, as illustrated in Figure 2.2, provides rotation through a power swivel, which are either powered by hydraulic or electric motors (Mitchell and Miska, 2011).

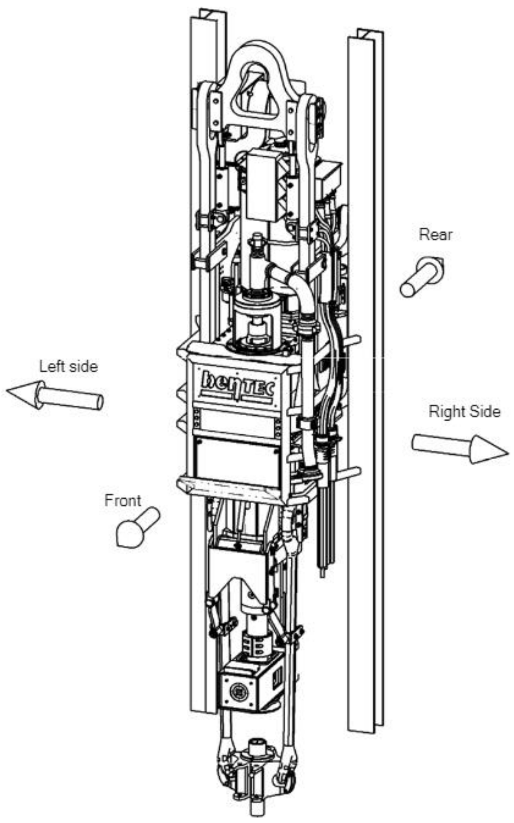

Figure 2.2: *Top Drive Schematic (Bentec, 2017)*

In addition to rotational motion, the swivel is also beneficial in other aspects, as it contributes to support the weight of the drillstring, and simultaneously allows mud flow from the circulation system to be pumped through the connected pipes. The swivel is attached to the travelling block and can run along vertical guide rails which extends from the crown block towards the rig floor (Assadi et al., 2019). To prevent influx and isolate high pressures from the bottom of the well, a hydraulic valve is also installed underneath the power swivel. Tools hanging below the swivel to unscrew connected pipes, such as a pipe handler and torque wrench, can also be observed.

Transition to topdrives in the oil and gas industry has offered several advantages with respect to drilling practices. Since the topdrive is connected to the hoisting system by the hook, it becomes a supplementary tool to the derrick, as it can move up and down along its structure. Consequently, longer lengths of drillpipe can be inserted into, or pulled out of the well, making the drilling process swifter. The biggest advantage from the topdrive however, is the fact that the swivel can be utilized to provide circulation of drilling fluid and torque while the pipe is either hoisted or lowered. With increased lengths of insertable pipe and circulation possibilities, the topdrive is able to maintain longer intervals of hole conditioning, which can indirectly reduce the frequency of stuck pipe incidences related to poor hole cleaning. (Assadi et al., 2019, Mitchell and Miska, 2011, Warren, 2010).

During a drilling operation, the topdrive applies torque to the drillstring, which in return transmit rotation to the bit, making the drillstring a natural part of the rotating system. The string is made up two main portions, the drillpipes and the bottom hole assembly (BHA). Figure 2.3 displays a drillstring and its typical components. However, one should note that the tool-composition of each portion is dependent of the specific drilling task that needs to be carried out.

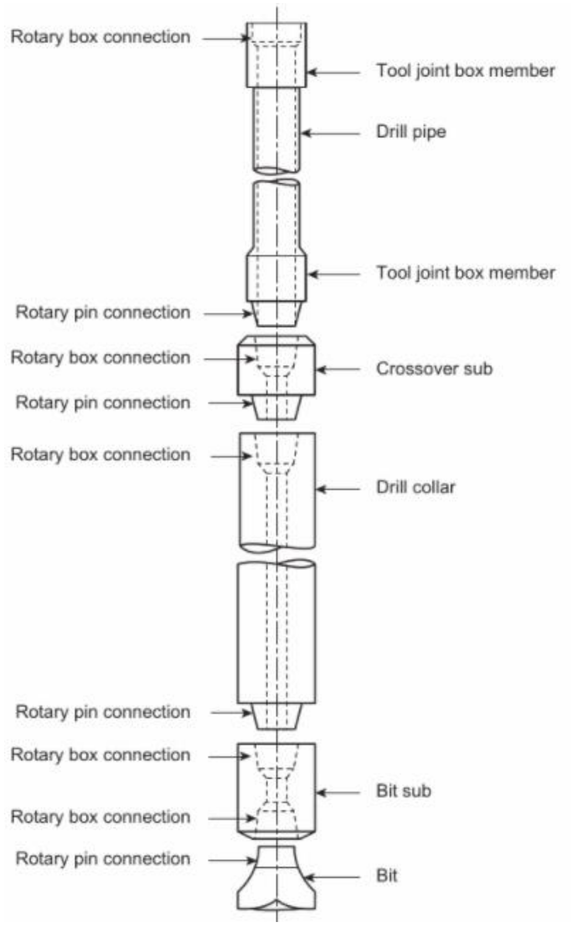

Figure 2.3: *Typical Components of a Drill String (Papavinasam, 2014)*

As mentioned in chapter 2.1.3, a drillpipe joint is commonly 30ft long and screwed together in lengths of three joints, called stands. This is done by means of tool joints, which can be located on each end of the drillpipe. The tool joint is divided into a male and female connector, called pin and box. The sectional compartment of the tool joint which the drillpipe is connected to has thicker walls than the remainder of the pipe, and is called upset. Upsets are necessary, and are often enhanced with tungsten carbide, as the point of connection is exposed to heavier loads and abrasive wear when the drillstring is rotated inside the wellbore (Assadi et al., 2019, Mitchell and Miska, 2011).

While the drillpipe assembly embraces the upper section of the drillstring, the BHA covers the lower part. Even though its composition will vary depending on the hole section that is drilled, the BHA is mainly composed of drill collars or heavy weight drill pipes (HWDP). In essence, drill collars are heavy, thick-walled steel tubulars. They are designed for a variety of purposes, such as ensuring efficient drilling by applying sufficient weight transfer on to the drillbit, providing stiffness in the BHA for directional control, and reducing fatigue induced failure loads such as buckling, by keeping the drillstring in tension (Alshaikh et al., 2018).

HWDP are steel tubulars with a width that is in-between that of the drill pipe and drill collar. This makes the HWDP capable of absorbing induced stresses that is being transferred from the drill collar, ultimately reducing the direct impact upon the drillpipe. In deviated wells, the HWDP serves a second purpose, as it is frequently used to replace the functionalities of the drill collar. This is because the width and weight of drill collars will increase the frictional forces, as well as the torque required to drill in inclined sections. Thus, it becomes more convenient to use HWDP in such geometries. Other supplementary components to the BHA that are generally used along with HWDP and drill collars are stabilizers, shock absorbers and drilling jars (Mitchell and Miska, 2011).

The drillbit is the last major component of the rotary system and can be located at the very bottom of the drillstring. Depending on the type of bit that is utilized, the applied torque from the topdrive can either be transferred into a circular crushing or cutting motion, which will crumble the formation as the drilling process ensues. At the bottom of the bit, jet nozzles are installed, as illustrated to the left on Figure 2.4. This is a typical nozzle-configuration for a roller cone bit, but the number of nozzles differs between specific bit-types and design. Nevertheless, implementation of nozzles is done in order to remove cuttings that accumulate underneath the bit, so that one can maintain the efficiency of the drilling operation. This is done by allowing passage of high-pressurized drilling fluid from the drillinstring through the bit. As the bit rotates, the ejected fluid will create enough turbulence to clean the face of the tool, and the removed sediments will be transferred out of the well via the circulation system (Assadi et al., 2019).

As of today, there are three types of bits that dominate the entire bit marked; the roller cone, the polycrystalline diamond cutter (PDC) and the hybrid bit. In terms of selecting the appropriate bit, there are some factors that need to be considered. These include, but are not limited to (Fjelde, 2019): formation hardness and abrasiveness, well profile, desired drilling speed, bit-durability, and steerability requirements. Therefore, selecting a specific bit design is heavily dependent on its area of application, as well as the weighted priorities of the operating company.

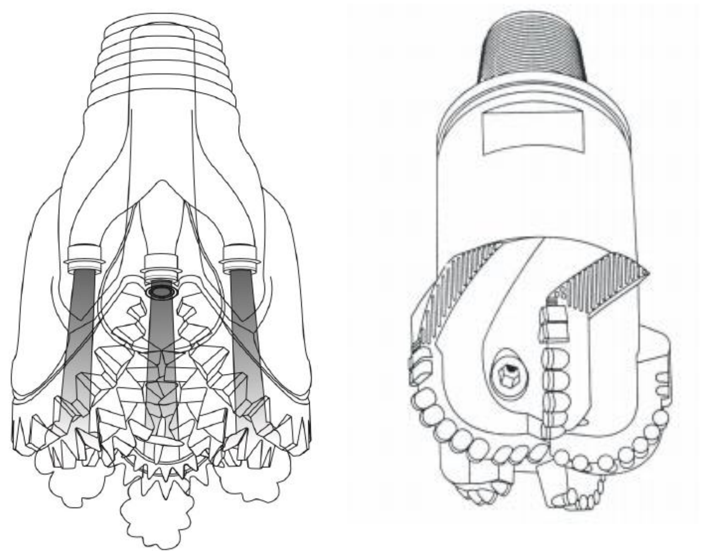
Figure 2.4: *Roller Cone Bit (Left) and PDC Bit (Right) (Assadi et al., 2019, Dagrain, 2001)*

2.1.5 Well Control System

A well control system is mainly implemented to prevent uncontrolled flow of formation fluids from the drilled well towards the surface. As the drillbit enters a permeable formation, which is a typical property of hydrocarbon prospects, the present pressure in the formation pore-space might be considerably larger than the pressure exerted by the hydrostatic fluid column of the drilling fluid. In such a scenario, fluids residing in the formation will enter the wellbore and force the drilling fluid out the hole. This is defined as a kick, and the term addresses any form of influx by oil, gas, or water into the well when there is drilling fluid present. The well control system allows the driller to (Mitchell and Miska, 2011):

- Detect a kick
- Closing the well at the surface
- Circulate the well under pressure to remove the formation fluids and increase the mud density
- Allow drillstring movement while the well is closed
- Divert a potential kick away from rig personnel and vital equipment

If the driller should fail to detect a kick, or there happens to be a malfunction in the well control system, a large influx of formation fluid into the wellbore might occur. This type of unwanted inflow is called a blowout. At worst, a blowout may lead to loss of equipment and lives, cause environmental damage, and huge losses of oil and gas reserves. Thus, a blowout is arguably the worst incident that can take place during a drilling operation. To prevent undesirable inflow of formation fluids, primary and secondary well control barriers are set in place during operational procedures.

Primary well control of a well is preserved by assuring that the mud column in the wellbore maintains a greater pressure than the formation that is being drilled through. In other words, the pressure differential between the drilling fluid and the formation needs to stay positive as drilling ensues (Mitchell and Miska, 2011). However, to large of a pressure differential can cause other issues, such as pipe sticking or fracture induced fluid loss. Primary control of the can be lost in two ways. The first occurs if the formation pressure in a zone that is drilled through turns out to be larger than the calculated values that was provided by the engineer. In this particular case, primary control will be lost as the designed mud-weight will be to light to withstand the formation pressure. The second case is if the pressure provided by the mud column suddenly decreases below the formation pressure. This can happen due to a decrease in the column-height of the drilling fluid, or due to a reduction in density, whereas both can occur due to a variety of circumstances (Assadi et al., 2019).

Secondary well control methods are required in cases where the primary control should fail, and the formation fluids start to flow into the borehole. The objective of a secondary control system is to hinder any influx into the wellbore, and eventually grant inflow to be circulated in a controlled manner to the surface so that it can be safely discharged, while simultaneously averting additional influx downhole (Assadi et al., 2019). This process is initiated by shutting the valves on the blowout preventer (BOP), so that one is able to close off the annular space at surface. The following step in this process is to circulate a denser mud down through the drillstring and up the annuls. The purpose of circulating a heavier mud would be to displace the influx while replacing the original mud and should prevent another influx from occurring as drilling is continued.

2.1.6 Well Monitoring System

During a drilling operation, there are several parameters that need close monitoring in order to optimize the drilling process. To ensure adequate adjustments and quick detection of drilling problems, it is always obligatory for the rig personnel to be able to monitor the operational progress. Thus, modern rigs have installed equipment that displays and simultaneously registers incoming data for most of the important variables associated with the drilling operation. Variables such as mud property, which are difficult to record automatically, will instead be measured, documented, and controlled continuously. Some of the most important parameters for the well monitoring system include, but are not limited to (Mitchell and Miska, 2011):

- Hookload (HKL)
- Rate of Penetration (ROP)
- Weight on Bit (WOB)
- Fluid Flow Rate
- Flow return
- Well Depth
- Pit Level
- Rotary Speed
- Rotary Torque
- Pump Pressure
- Pump Rate
- Fluid Properties (e.g. density, viscosity, gas and sand content salinity, solids content, temperature)

Together with consistent historical data for comparable operations, these recorded variables will offer aid to the driller in terms of predicting and noticing probable drilling hazards. Monitoring the mud system is an especially vital job that must be accomplished to sustain sufficient well control, as mud have good indicating properties related to drilling problems such as circulation loss and kick detection (Mitchell and Miska, 2011). By maintaining an appropriate monitoring frequency, the impact of aforementioned incidences can be kept to a minimum.

Nowadays, centralized well monitor systems are often carried on modern rig sites. Thus, if the situation should call for it, the monitoring system can be housed in the engineer's office or in the geologist's office at the rig site. The installed monitoring units grants access to detailed information about the fluids that are circulated, as well as the formations that are being drilled through. This sort of information provided by the well monitoring system makes it possible to create varying analysis methods to detect the presence of oil or gas reservoirs.

Lastly, well monitoring systems also include installment of downhole tools to improve efficiency of direction control in deviated wells. These are called measurement-while-drilling (MWD) tools and are run together with the BHA to continuously send input regarding well position and well trajectory to the surface. These tools normally provide information through use of mud pulses, which sends pressure pulses to the surface through the drilling fluid that is being contained in the drillstring. At surface level, the received pulses are interpreted through encoding and different signal-processing methods (Schlumberger, 2021e).

2.2 Drilling a Well

If possible, the drilling process would have been completed in one run all the way down to the reservoir. However, drilling a single hole section is not possible in practice due to issues related to the geology of existing subsurface layers, as well as formation pressures that is encountered during drilling (Assadi et al., 2019). Furthermore, large diameter holes are costly as they require more amount of steel to create the inserted tubulars, are time consuming to drill and increase the chances of hole collapse. Thus, a typical well is drilled section by section, where each subsequent interval is smaller in diameter than the previous, utilizing a casing, which essentially is the same as a hollow steel pipe, in each section to isolate the problematic formations once they have been breached. How fast each section is drilled will depend on the ROP, formation characteristics and circulation time, whereas the last one in particular consumes a considerable amount of time with respect to cleaning operations.

A typical offshore well can be seen in Figure 2.5, where each section, as well as their dimensions, is illustrated. Although the dimensions may vary in other parts of the world, these are the numbers that are typically utilized in the North Sea and will therefore be used as examples in the subsequent chapters to describe the drilling process.

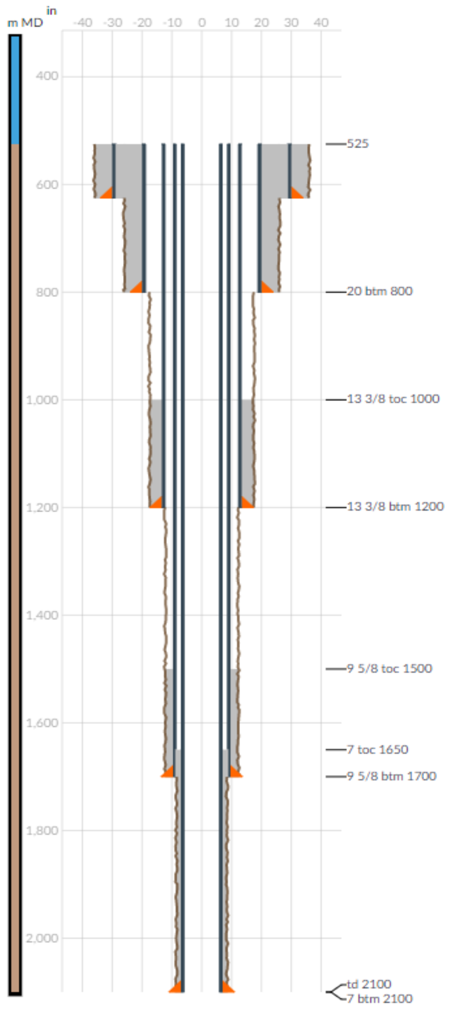

Figure 2.5: *Typical Casing Design, made in Oliasoft WellDesign Software.*

2.2.1 Setting the Conductor

To be able to reach the desired depth of each drilled interval, the drillstring needs to be elongated in a controlled manner, which is done by adding lengths of standpipes as the drilling progresses. Once the stand is ready, circulation of drilling fluid is halted, and the drillstring is detached from the topdrive and held in place onto the deck using a clamping device called slips. The stand is then screwed onto the bottom of the topdrive, as well as the top of the entire drillstring (Schlumberger, 2021d). Once both connections are done, the slips are released, and drilling can be continued. This particular process is termed as "making a connection", and consecutive connections that are made is called tripping. Tripping can be done both in and out of the well, where entering often is referred to as run in hole (RIH), while exiting in many cases is referred to as pull out of hole (POOH).

The first step in a drilling operation is to run a large diameter casing, called conductor, into the formation. A conductor is the common name for the first casing that is installed during drilling, and its main purpose is to prevent unconsolidated layers, such as sand, from collapsing into the well when drilling is continued. In addition, the conductor should be able to support the weight of the wellhead when it is installed later in the drilling process (Aadnoy, 2011).

If the formation on the seabed is soft, the conductor is often just hammered with a hydraulic hammering tool or jetted into the ground. Jetting refers to the process where high pressure sea water is pumped through the drillpipe to lift the sediments on the seabed up and out of the conductor (Schlumberger, 2021c). However, if the formation is hard, drilling of a hole with outer diameter (OD) of 36" is initiated. Once the hole has been drilled, the drillpipe is pulled out of the hole and the conductor, which normally has an OD of 30" is installed.

After installing the conductor, the drillpipe is pulled back up, and cementing tools are lowered to the ocean floor to pump cement into the annular space between the conductor and wellbore. Cement pumping during drilling is done to strengthen the casing and its sealing capabilities, so that fluid migration between formations is prevented, and to reduce the chances of formation cavings (Assadi et al., 2019). The last part of the process is trip out the drillstring that holds the cementing tools and once the cement has hardened, preparations for the next operation ensues.

2.2.2 Drilling and Casing the 26" Hole Section

The first hole section is drilled with a drillbit through the conductor casing. In other words, the drillbit used is required to have a smaller diameter than the inner diameter (ID) of the conductor. The ID of the conductor is roughly 28", so a drillbit of 26" is typically used for this hole section (Assadi et al., 2019). As the hole section is drilled, drilling fluid is circulated down through nozzles at the drillbit, and up the annular space between the drillpipe and the wellbore, to ensure that drilled cuttings from the bit face is carried all the way to the surface. Transported cuttings is sent through vibrating shakers, which are machines with sieves of varying mesh size, to separate the solids from the mud before the fluid is circulated back down through the drillpipe, and the transportation process of cuttings is repeated.

When the bit reaches desired target depth, the drillstring is pulled out of the hole, and the next steel tubular, the surface casing, is run in hole. The surface casing typically has a 20" OD and is delivered to the rig in joints (Assadi et al., 2019). Casing joints usually has a standard length of 40ft and has threaded connections at either ends (Schlumberger, 2021a). Each joint of the surface casing is assembled to a specified length, called a casing string, before it is submerged to the bottom of the drilled interval.

The last process in the 26” hole section would be to pump cement slurry into the annular space between casing and wellbore. In most cases, the surface casing is cemented all the way up to the seabed. This is done in order to provide enough structural integrity to the well, so that it is able to withstand the weight of the wellhead as it is installed on top of the surface casing. After successful cementation and testing of the hole section, the drilling crew can move on to the next phase. In offshore drilling procedures, a typical well is almost never drilled purely vertical. The most common practice is to start drilling in an inclined path towards the reservoir. For the 26” hole section, the inclination will be increased as drilling progresses and is often called a build-up section. If the build-up section is not initiated in the 26” section, it will begin in the 17 ½” hole section instead.

2.2.3 Drilling and Casing the 17 ½” Hole Section

Before the rig crew continue to drill the second hole section, the wellhead housing and blowout preventer (BOP) need to be installed. The wellhead housing is installed on top of the last casing joint in order to support weight of the BOP and the subsequent casings (Crumpton, 2018), while the BOP stack is installed as there is a high chance of encountering pressurized fluids from this point onward . If the drilling crew were to come across high pressurized fluids without the BOP, hydrocarbon sources would start to flow uncontrollably to the surface, causing a blowout (Assadi et al., 2019).

The drilling operation can be continued once the BOP is installed and successfully pressure tested. The second hole section is usually drilled with a 17 ½” bit down to target depth, and the drillpipe is now mobilized through the BOP stack. When target depth is reached, the drillstring is pulled out, and a 13 ⅜” steel tubular is inserted into the 17 ½” hole section (Aadnoy, 2011). This is the intermediate casing, which is RIH and cemented to a minimum depth of 200 meters above the bottom (shoe) of the tubular.

After the first- and second hole section, the well has usually been drilled close to the vicinity of the hydrocarbon reservoir. However, North Sea wells are often deeper and more challenging to drill, as one need to surpass an added number of sedimentary layers. Thus, common practice often require installation of multiple intermediate string sections, making this particular section the longest to drill (Crumpton, 2018, Aadnoy, 2011). Alternatively, consecutive intermediate casings might be replaced with liner strings. A liner is a steel tubular similar to casings, except that they are fixed at the bottom of the previous tubular that is installed, in contrast to regular casings that extend all the way back to the wellhead. This makes them cheaper and easier to access than regular casings and is a promising option in wells where accessibility is a problem.

2.2.4 Drilling and Casing the 12 ¼” Hole Section

The third hole section is drilled towards the reservoir cap rock with a 12 ¼” bit, and progression is halted when the bit is located a couple of meters above the reservoir section. In order to move on with the drilling operation the production casing, a 9 ⅝” steel tubular, is installed in the 12 ¼” hole section. The production casing serves two main purposes, where the first one it to allow modifications of production equipment if deemed necessary. The second purpose of this particular casing is to act as a barrier element during production. Thus, the 9 ⅝” is set in an impermeable formation such as shale, to minimize the risk of leakage from a permeable reservoir rock (Torbergsen et al., 2012) The hole section is then completed by cementing the 9 ⅝” casing to a minimum depth of 200 meters above the casing shoe. As the cements hardens, the rig crew can start preparations for their last drilling procedure, which is to access the hydrocarbon reservoir.

For offshore wells, the 12 ¼” hole is often drilled as a hold-section, which means that the inclination angle in the drilling direction is kept constant as drilling proceeds. This type of profile is defined as a slanted profile. Depending on the planned well profile, another build-section might be initiated in this section if one wishes to enter the reservoir horizontally. Lastly, the well might be designed with a drop off section. A drop off section is when the inclination is gradually reduced so that one is able to enter the reservoir vertically, which is typical for an S-shaped well.

2.2.5 Drilling the 8 ½” hole section

To enter the reservoir, an 8 ½” OD hole is drilled through the 9 ⅝” casing. As drilling ensues, signs of hydrocarbon resources would start to emerge, where the most obvious indication would be oil presence on transported cutting. If gas is present, it would be transported through the drilling fluid and trigger gas detecting sensors that are installed to the of the drilling platform (Assadi et al., 2019).

When drilling has been performed across the whole length of the reservoir, a lower completion of the reservoir follows. This particular section is usually completed in two different ways, namely open hole, or cased and perforated. However, both completion methods require use of sand screens in reservoirs where sand production is to be expected (Crumpton, 2018).

An open hole completion, as the name indicates, is when the hole section is left open in order to allow hydrocarbons to flow freely from the reservoir and into the well. Hence, no cementing is needed in the open hole completion. This method is mostly applied in areas where there is no need for zonal isolation (Bellarby, 2009). Other variations of the open hole completion are executed using a predrilled/slotted liner, or with sand screens. The first variation typically exploits a 7” OD liner that is hung and cemented onto the 9 ⅝” casing shoe and into the open hole. Since the liner is slotted, it will act as a conduit for the hydrocarbons that are flowing from the reservoir.

Cased and perforated completions is performed by running a liner into the 8 ½” hole section. After the tubular is set, cement is pumped into the annular space between the liner and wellbore. Once the cement has hardened; small, explosive charges are used to create channels that connect the well to the reservoir (Crumpton, 2018). In addition to zonal isolation, cased and perforated completions perform considerably better than open hole completion in terms of production, and are therefore extensively utilized in the North Sea and other offshore areas (Bellarby, 2009).

When either of the completion methods are set, a retrievable pipe with an OD larger or equal to the production liner is installed. This tubular is known as the production tubing, and is run from the top of the wellhead through the 9 ⅝” casing to create a path for the produced hydrocarbons to flow through (Crumpton, 2018). After the production tubing is installed, the drilling process is considered finished, and oil or gas production can be initiated.

3 Stuck Pipe

Pipe sticking during drilling operations is a common phenomenon and is known to occur in varying degrees of severity which can be triggered by several factors. These variables consist of, but are not limited to: well profile, drilling mud properties, pore pressure, formation lithology, BHA configuration and rig crew experience (Alshaikh et al., 2018). Also, chances of successfully freeing the pipe decreases over time. As a matter of fact, only 50% of pipe sticking incidences are possible to exempt within 4 hours, while less than 10% of the incidences are resolved after 4 hours (Mitchell, 2011). In addition, incorrect ways of approaching the problem could exacerbate the stuck pipe condition and should be avoided if possible.

Due to the abovementioned statements, drilling operators and contractors must make quick and informed decisions in order to reduce or alleviate an ongoing sticking scenario. The first step would be to identify the stuck pipe situation and the mechanisms that are instigating a sticking condition. These mechanisms are classified based on the force that inhibits pipe movement, and are often divided into three main categories, namely:

1. Differential Sticking
2. Pack-off /Bridging:
   - Pack-off: Restrictions caused by small sized particles, which clogs the annular space between drillstring and wellbore.
   - Bridging: Restrictions in drillstring movement caused by a blockage of medium to large sized debris getting wedged across the drillstring.
3. Wellbore Geometry

There are several papers and books that define these classes and describes the underlying mechanisms for sticking thoroughly. Therefore, the following subchapters describing each of the main categories will take basis in readings from Bowes and Procter (1997), Mitchell (2011) Alshaikh et al. (2018), Glomstad (2012), Strand (2014), Gulsrud et al. (2009), Petropedia and Schlumberger Oilfield Glossary.

3.1 Differential Sticking

Differential sticking is defined as an "imbalance between the hydrostatic pressure in the wellbore and the pore pressure of a permeable formation" (Bowes and Procter, 1997). It is a time reliant phenomenon which takes place when static conditions are present and triggers complications in the transition towards dynamic conditions. More specifically, the sticking mechanism occurs when the hydrostatic pressure of the drilling mud becomes much greater than the formation pore pressure. The transpiring pressure difference is labeled as overbalance, and the ensuing force of the overbalance acting on a surface-area of the drillstring, in combination with slow or stationary pipe movement, is what mainly initiates a differential sticking incident. As a result, the drillstring is embedded in the filter cake and becomes stuck. The force causing pipe sticking is determined as a function of the drill collar contact area A_c, the differential pressure ΔP, and the friction coefficient μ (Bourgoyne et al., 1986):

$$F = \mu \times \Delta P \times A_c \tag{3.1}$$

Where ΔP can be expressed as the difference between the hydrostatic pressure of the mud column P_M and the formation pore pressure P_{FF} :

$$\Delta P = P_M - P_{FF} \tag{3.2}$$

In addition to no string movement and high overbalance, awareness of the drilling mud is of importance when identifying differential sticking. Since there are no physical hindrance in the wellbore other than the drillpipe sticking to the bore hole wall, full and unrestricted circulation of drilling fluid will still be possible (Bowes and Procter, 1997).

3.2 Pack-off/Bridging

Bridging and pack-off happens when portions of the wellbore wall cave-in around the drillstring, or when the wellbore around the drillstring is clogged with varying sizes of solids, inhibiting axial or lateral pipe movement, and preventing or limiting circulation. Chances that this particular type of sticking mechanism can ensue is equally viable when making connections or tripping, and is regarded to be the most occurring and severe form of pipe sticking that can take place (Mitchell, 2011). There are a wide range of causes that may lead to pack-off or bridging induced pipe sticking, and the most common reasons, as well as proactive and preventative measurements will be discussed below.

3.2.1 Poor Hole Cleaning

Pipe sticking due to poor hole cleaning is a result of cuttings or cavings that fail to stay suspended in the circulation fluid. In terms of hole cleaning, circulation fluids need to have the correct viscosity and appropriate flow rate in order to efficiently transport the cuttings out of the wellbore. If either is lacking, the lifting force caused by drag is overcome by the force of gravity, and the carrying capacity of the fluid becomes insubstantial.

Inadequate hole cleaning is an issue that is rarely encountered during vertical drilling, because the distance to bottom of the well is equal to the total depth of the drilled interval. This creates a bigger window for the particles to stay suspended and eases the transportation process done by the drilling fluid. In inclined and horizontal sections however, the low side of the well becomes the width of the hole from its point of center. Consequently, the travelling distance for the suspended particles to the bottom of the well becomes drastically shorter. Due to the decline in height provided by the well, particles will now accumulate more rapidly on the low side of the drillpipe (Cayeux et al., 2014). Over time, these particles will develop into layers called cutting beds and cause a pack-off.

In inclined well sections an additional issue is of concern, as accumulated cutting beds are prone to avalanching effects. Avalanching occurs as a consequence of having insufficient flow rates, which allows the height of the formed cutting beds to increase. Once the cutting bed reaches a certain height, the gravitational force acting on the solids will become larger than the upwards force created by the flow rate. As a result, settled beds will become unstable and start to slide along the inclined plane, causing an instantaneous build of sediments around the drillpipe (Cayeux et al., 2014). Due to the fact that the drillpipe becomes packed-off in a short amount of time, avalanching is considered as the worst out of the different inadequate hole cleaning scenarios.

Parameters that might indicate stuck pipe due to poor hole cleaning are:

1) Increased torque and drag (Doshi, 2014)
2) Increase in HKL may be experienced while POOH (Bowes and Procter, 1997)
3) Erratic pump pressure (Bowes and Procter, 1997)

4) Poor weight transfer to drillbit, which causes smaller than expected HKL values (Bowes and Procter, 1997)

3.2.2 Unconsolidated formations

In contrast to consolidated formations, unconsolidated formations have a foundation built up by particle-mixtures which lacks strong, adherent bonding between the existing grains or rocks. This attribute makes them easy to un-bond and is often associated with materials such as sand or gravel. When drilling through an unconsolidated formation, the supporting rock of the foundation is gradually removed. Due to weak particle-bonding between the grains, the fluid column intended to withstand the formation pressure will start to seep into the unconsolidated formation, breaking it apart. The hydrostatic overbalance present will unable to support the formations that is being drilled through and formation integrity is lost, causing sand or gravel to collapse into the hole. This will lead to the drilstring being packed-off, and the pipe becomes stuck. Surface parameters that might indicate pipe sticking due to unconsolidated formations are:

1) Increase of mud weight, as unconsolidated sediments mix with the drilling mud (Doshi, 2014)
2) Increase in pump pressure (Bowes and Procter, 1997)
3) Increase in torque when drilling (Doshi, 2014)
4) Abnormal drag observed during pick-up (Doshi, 2014)

3.2.3 Fractured & Faulted Formations

A naturally fractured rock-system can often be located near the vicinity of faults. Rocks near faults are exposed to large stresses, which can generate failure modes that breaks it into pieces of varying sizes. If the pieces happen to be loosely packed, they can fall into the wellbore and wedge themselves across the drillstring, causing the string to get jammed. This type of sticking can be identified through:

1) Sudden changes in torque and drag, with erratic patterns during drilling (Doshi, 2014)
2) Circulation may be restricted (Doshi, 2014)
3) Sticking might be instantaneous (Bowes and Procter, 1997)

3.2.4 Naturally Over-Pressured Shale Collapse

Sediments can in some cases exist naturally in the subsurface with a pore pressure that is abnormally high. This happens in areas when fluid-filled sediments, such as shale, is buried so quick that the pore fluids inside the sediments become trapped. Shale in particular, is known to be a rock type with high porosity and little to no permeability. This attribute causes the water inside the pore space to be compressed and pressurized. As the overburden pressure increases, pressure levels inside the formation exceeds the expected collapse pressure for a given depth due to entrapment of pore fluids. In such stratigraphic layers, the normal hydrostatic pressure gradient will be significantly lower than the formation pore pressure of shale. If over-pressured shale is overlooked and inadequate mud weights are utilized, the wellbore will become unstable and collapse. Rig site indications for pipe sticking caused by naturally over-pressure shale are (Bowes and Procter, 1997):

1) Increased torque and drag.
2) Circulation is impossible or restricted.

3) Increased ROP

3.2.5 Induced Over-Pressured Shale Collapse

Shale formations can become problematic due to its absorption properties. If water-based mud (WBM) is used during drilling, water residing in the drilling mud will start to seep into the shale formations. Over time, water seepage will increase the pore pressure of the rock and instigate a shift in the pressure regime of the exposed formation. If exposure to shale formations is continuing over the course of days, with no increase or reduction in the hydrostatic wellbore pressure, the shale formation will attain a greater internal pressure than the bore hole. This will result in a wellbore collapse, as the particle-bonding of the shale will lose its adherence due to the increase of pore fluids, and a situation comparable to naturally over-pressured shale will follow. Surface indications for pipe sticking related to induced over-pressured shale are (Bowes and Procter, 1997):

1) Increased torque and drag.
2) Circulation is limited or impossible.

3.2.6 Reactive Formations

This type of pipe sticking incident is typical in young clay and shale formations, which are water sensitive. If a drilling mud with inadequate inhibiting characteristics, such as lack of CaCl and KCl additives, is utilized throughout such formations, swelling occurs due to chemical interaction with the drilling mud. The chemical reaction between formation and mud is time dependent. Thus, tempo of swelling may vary from a couple of hours to several days. When swelling of sedimentary layers finally takes place, it will narrow the wellbore-space, causing the pipe to stick. Indications of reactive formations are (Bowes and Procter, 1997):

1) Circulation of drilling mud is highly restricted or impossible
2) Changes in ECD because of increase in low-gravity solids (LGS)
3) Fluctuations or an increase in pump pressure
4) Typically occurs when POOH

3.2.7 Tectonically Stressed Formations

Pipe sticking due to tectonically stressed formations is a phenomenon that often occur in or near mountainous regions. Tectonic stresses build up in these areas, as movement in the earth's crust forces the rock to compress or elongate. If the rock is under compression, pressure caused by tectonic plate movement will make the rock buckle. Buckling of sedimentary rocks lead to an increase of formation pressure in the tectonically stressed area. Drilling through a such an area will lead to rock fragments collapsing around the drillstring and bridge, as the formation pressure is significantly higher compared to other exposed formations in the wellbore. To prevent the bore hole wall from collapsing, the required hydrostatic pressure needs to be adjusted. However, for tectonically stressed formations, the adjustment might be so large that it exceeds the fracture pressure of pre-drilled formations, leading to fracturing of overlying formations instead. Indications of tectonically stressed formations are (Bowes and Procter, 1997) :

1) Increase in return of cutting volumes at shakers in relation to the volume of hole that is drilled
2) Circulation is restricted or non-existent if the pipe is stuck

3) Torque and drag increase observed

3.2.8 Cement Blocks/Junk

Lastly, the drillstring can become stuck if cement blocks or junk falls into the well and jams the wellbore. For cement, the mechanisms can either occur when hard cement around the casing shoe, openhole squeeze plug, or kick-off plug becomes unstable due to insufficient curing time, while junk often accumulates in the wellbore if there happens to be downhole equipment failure, or if the rig crew forgets to install a hole cover. Rig site indications for sticking caused by cement or junk are (Bowes and Procter, 1997):

1) Torque becomes erratic
2) Circulation remains unrestricted
3) Rotation and downward movement may still be possible

3.3 Wellbore Geometry

Pipe sticking related to wellbore geometry happens in cases where there is a conflict between the configuration of bottom hole assembly (BHA) and the shape of the wellbore. Specifically, it occurs due to misconfigurations in the wellbore shape. As the BHA travels into the misconfigured area, movement becomes restricted in upward or downward direction, and the pipe becomes stuck. Similar to pack-off/bridging, there are several factors that may lead to cases of stuck pipe which are related to wellbore geometry. The most common problems related to wellbore geometry will briefly be discussed below.

3.3.1 Undergauge Hole

Undergauge hole resembles the effects of mobile formations, but is mainly a product of drilling through hard, abrasive rocks. Pipe sticking due to undergauge hole can also occur after coring, or when a roller cone bit is run and followed by a PDC bit. Drilling abrasive rock-types wears down the bit-gauge protection of the drill bit, which results in a smaller wellbore diameter than intended. When the next bit is run in hole (RIH), it faces resistance when it comes in contact with the undergauge section of the bore hole. If the drillstring is tripped rapidly into the wellbore without reaming, there is a high possibility that the descending bit gets jammed in the undergauge section of the bore hole. Rig site indications of an undergauge hole section are (Bowes and Procter, 1997):

1) Decrease in HKL, as increase in set down weight will be experienced
2) Pipe only gets stuck when RIH
3) Bit is stuck close to the bottom or at the top of a cored section
4) Circulation is still possible or only slightly restricted

3.3.2 Key Seating

When torsion and strong side wall forces is present during a drilling operation, the drill pipe might rotate against a single point of a rock-surface for a long period of time. The force from the rotating bit will cause abrasion on the bore hole wall, forming a groove or key seat in the wellbore. As the drillstring is tripped out past the point where the key seat is formed, parts of the string that is bending against the bore hole will straighten, and the BHA or the tool joints can be pulled into the groove, wedge itself and become stuck. This particular mechanism usually occurs in medium-soft to medium-hard formations when there are sudden deviations in wellbore direction or angle, after long periods of drilling without wiper trips through the dogleg section, while POOH, or in wells where high side wall forces and string rotation is present. Surface indications for key seating are (Bowes and Procter, 1997):

1) Abrupt increase of HKL when pulling out of the hole as BHA reaches depth of dogleg
2) Sticking only happens when POOH
3) Circulation is unrestricted

3.3.3 Doglegs and Ledges

Sticking due to doglegs or ledges are caused by deformation or unwanted changes of the wellbore trajectory. For a ledge to develop, there needs to be varying types of formations present. The variety of rock types and firmness of sediments will create ledges at the formation-boundary as the wellbore passes through. Doglegs are abrupt changes in the wellbore direction, which are caused by the bit being deflected due to the rock characteristics present in the formation. Similarly, sudden deviations in bore hole angle can cause a kink in the direction of the well when a directional BHA is drilled. Directional changes, POOH, drilling through hard and soft layers that are interbedded or utilization of an unsuitable BHA are all elements that can lead to dogleg and ledge induced sticking. It is also worth mentioning that doglegs are one of the major contributing factors to wellbore geometry sticking, as they can trigger other sticking scenarios such as drillstring failure from high side loads and torque, and key seating. Doglegs also increase the possibility of provoking other types of sticking mechanisms related to differential sticking or pack-off/bridging, making them quite troublesome. Rig site indications of doglegs and ledges are (Bowes and Procter, 1997):

1) HKL suddenly becomes erratic
2) Problems take place at a fixed depth
3) Circulation is possible

3.3.4 Mobile Formations

Pipe sticking due to mobile formations is a result of compression caused by overburden forces, and can often be observed in salt formations or creeping shales. The compressional force deforms the mobile layers plastically, as the mud weight in the well is deemed insufficient to prevent the formation from moving into bore hole. Mobile formations reduce the diameter of the wellbore to a point where it is smaller than the nominal diameter. This will create problems running logging tools, casing and BHA. At worst, the pipe could become stuck. Surface parameters that might indicate pipe sticking for mobile formations are (Bowes and Procter, 1997):

1) Increase in HKL when POOH, and decrease in HKL when RIH
2) BHA sticks at mobile formation depth
3) Circulation is restricted when BHA is at mobile formation depth

4 Machine Learning and Artificial Neural Networks

In terms of machine learning, engineers often lack the required data science knowledge necessary to comprehend the topic. This lack of knowledge can often lead to doubts regarding the outputs from a ML model, as many engineers consider ML as a system where information is taken and processed in a "black box", which then yields a result. Thus, the following sub-chapters have been added to contribute to an overall understanding of ML models and how they are categorized, as well as to give insight in how a typical workflow is conducted for a ML project.

4.1 Machine Learning Basics

Machine learning is a subset of artificial intelligence. It is defined as a collection of various algorithms used to teach computers pattern recognition in data to be used for future forecasting and prediction or as a quality controlling measurement for performance optimization (Belyadi and Haghighat, 2021). ML offers computers the ability to acquire relevant information without being explicitly programmed. More precisely, they do not require specific machine characteristics nor application of physical laws to function. ML models are therefore able to determine dependency of existing variables through the provided data, and only this data.

However, there are still an existing need for a human assistance in order for a ML model to function properly. Model assistance from humans are essential, but the methodology in which this expertise is provided is dissimilar to that of a physics-based model. In machine learning, domain knowledge is particularly supplied in four ways (Bangert, 2021):

1) Proving all relevant variables and excluding all irrelevant variables. This type of activity may also include some elementary processed variables from feature engineering. For example, if one considers the relation between two variables as important, it may be feasible to apply the ratio between these variables as a separate column in the data table.
2) Explicitly adding any essential restrictions that must be obeyed.
3) Providing empirical data that is significant and representative of the situation.
4) Assessing the results of candidate models to ensure that the output is behaving as expected.

The abovementioned inputs are of importance but are easily applied by an engineer or domain expert who is familiar with the topic of relevance. If the expert happens to be aware of the requirements from a data science perspective as well, this process should run rather smoothly.

According to Bangert (2021), the topic of ML is divided into three parts. The first one consists of a variety of standardized, characteristic models that could be utilized onto the data present. These models go under names such as k-means clustering, random forest (RF), gradient boosting (GB), and artificial neural networks (ANNs). Secondly, each of these standardized models include several prescriptions which are called algorithms. In simple terms, an algorithm can be defined as a step-by-step procedure which informs the user on how model coefficients can be calculated from a dataset. In an ML environment, utilization of these characteristic models is commonly known as the training phase. After training is conducted, the initial standardized model has been converted to a model that is able to handle the specific dataset that has been provided. Lastly, the final model needs to be deployed for usage purposes. This stage is usually named test or evaluation phase and are often considered less complicated and swifter to handle than the training process. Once the model is trained, the model is fully applicable in real time, which is one of the key features of ML that makes it convincingly beneficial. However, one still need to emphasize the importance of enclosing the model into the right infrastructure, as failing to do so will heavily impair its performance.

With respect to machine learning, there are two essential subjects that are focused on from a data science perspective (Bangert, 2021, Belyadi and Haghighat, 2021). The first point of focus is to sufficiently prepare the data before the learning phase. The second point of focus is to appropriately test the final model, and its effectiveness must be verified through accurate mathematical methodologies. These pre and post processing operations are applied to ML models in order to capture the scientific portions of the data science project. In addition to these scientific portions, there are also bureaucratic and regulatory sections that concern data collection, as well as handling stakeholders' requirements regarding the application.

4.2 Workflow

As the oil and gas (O&G) industry is progressing towards a more digitalized sector, the vast number of existing engineers in this field is required to widen their scope of knowledge surrounding the field of data science. This also includes the topic of ML, as the key to successfully develop and implement a ML project relies on a groups capability to combine domain knowledge with data science expertise. Thus, this subchapter will focus on describing a typical workflow that can be applied to a variety ML projects, and will be heavily influenced by the book of Belyadi and Haghighat (2021), named *Machine Learning Guide for Oil and Gas Using Python: A Step-by-Step Breakdown with Data, Algorithms, Codes, and Applications*, and the book *Machine Learning and Data Science in the Oil and Gas Industry: Best Practices, Tools and Case Studies*, written by Bangert (2021)

4.2.1 Data Gathering and Integration

Data gathering is considered as one of the most crucial steps when working with a ML project, as many ML projects end up as unsuccessful due to the absence of available data. Thus, focus on data availability within an organization is an important factor to consider before starting any ML project. If the desired data are found challenging to access or just unavailable, the best recommendation would be to divert the attention towards projects with available data, which also have a high probability of providing concrete results that can validate the potential of ML within said organization.

Belyadi and Haghighat (2021) mentions that data scientists can spend up to 80% of their time collecting data, and only spend the remaining 20% to analyze it. A great tool for shortening the time spent on data gathering would be the implementation of centralized data warehouses, such as SQL, to store the data. This would remove the necessity of visiting multiple sources to gather the required information to perform a ML analysis.

4.2.2 Data Cleaning

Data cleaning is the second process after the data has been gathered and integrated. Within the field of ML, lack of adequate time spent on interpreting and cleaning the data before applying the ML algorithms is one of the leading reasons for failure. For data cleaning of any ML model, data visualization, outlier detection, and data imputation are the three main procedures which are frequently used to properly pre-process the data (Belyadi and Haghighat, 2021).

4.2.2.1 Data Visualization:

Data visualization is the first step in terms of data cleaning, as visualization grants an easy overlook to detect any anomalies or outliers which are present in the received data. Presence of bad or undesirable data might yield unexpected and flawed results, which can easily invalidate a model. Hence, detecting these points by the aid of visualization packages becomes of essence before starting an analysis. In addition, data visualization can also contribute to attain a better understanding of how each feature in the dataset is distributed.

4.2.2.2 Outlier Detection:

As mentioned earlier, visualizing the data at hand provides good guidance in detecting outliers. In addition, one can look at the existing parameters through specific visualization methods such as heat maps, box plots, scatter, and distribution plots to gain a better understanding of the values that are present. This is done to ensure that the values in the dataset are within a reasonable range. For instance, if we have a dataset where the ROP generally lies around 100 and 200 ft/h, and we suddenly have a couple of datapoints which have values of 800 ft/h, it is important to investigate those anomalous points. If the validity of the outliers can be confirmed, the points are to be kept. If the opposite holds true, one should remove the troublesome points before continuing. Furthermore, many library packages such as Pythons Pandas offers functions to easily access basic statistics for the data that is being handled. These statistics often include the minimum, maximum, frequency, median, average and standard deviation of the dataset, and should be obtained before and after the data has been cleaned to assure that the final range of parameter-values are considered reasonable.

4.2.2.3 Data Imputation:

Lastly, we have data imputation, which refers to the various approaches used to handle missing data in a dataset. These values often appear in the dataset as N/A, NaN, -999.99 or similar, and is an issue of concern that needs to be dealt with. There are a couple of approaches to handle this problem. The first one is to simply remove any row in the dataset which contains NaN values or similar and is the easiest way of treating missing data. However, the downside of this particular method is that the number of samples could drastically diminish as a consequence of removing large amounts of samples with missing information.

The second approach is to replace missing values with the median or mean values of the row that is above and below the missing datapoint. This procedure is rather easy and uncomplicated to perform. Nonetheless, introducing erroneous data to a dataset can create additional uncertainty. Hence, the better option might be to solely remove the rows of missing data, unless there exist evidence to justify that this method will yield accurate results.

The third and last approach for filling missing values is to use algorithms or packages that exists in the coding language for this specific purpose. In Python, the most commonly used package and algorithms to fill in missing values are the Fancyimpute package, Multivariate Imputation by Chained Equations and k-nearest neighbor (KNN). These algorithms work greatly when certain conditions are met, such as presence of a characteristic relationship between the existing parameters and the ones that are missing. For instance, if information regarding “Mud Density in” in one well is partially lacking, but “Mud Density Out” for the same well is available, these algorithms might produce meaningful answers. The largest challenge associated with this in terms of O&G operations are the fact that when there is absence of well data with respect to one parameter, there is a high probability that most of the remaining data is missing as well. Therefore, it is not recommended to take advantage of these packages if a large portion of the parameters are simply missing from the data, because this introduces that chance of producing flawed results.

4.2.3 Feature Ranking/Selection

Followed by data cleaning, the next step is feature ranking and selection. In short, feature selection is the process of reducing the number of input variables (features) when developing a predictive model, while feature ranking is the process of determining the level of impact that each of the selected input variable will have on the predictive model. When doing feature selection and ranking towards a ML project, one must emphasize the importance of having both domain knowledge and AI expertise present. If a data scientist has little to no experience within the industry of interest, it becomes immensely difficult for that person to understand the physical relationships between the variables, as well as the impact that each parameter has on the data that is being analyzed. This is where the domain expert comes in, as their knowledge within a specific disciplinary is invaluable.

For feature ranking and selection, the first step is to identify the problem the data scientist must solve. If the problem is related to completion operations, the next logical step is to get hold of an experienced completion engineer and pair this person with the data scientist. This is done because the completion engineer knows all details surrounding the topic at hand, and their expertise will be of great benefit to understand the data that is going to be worked on. On the other hand, the data scientist has a thorough knowledge regarding the algorithms and the underlying statistics behind them. Thus, feature selection is tied directly with domain expertise.

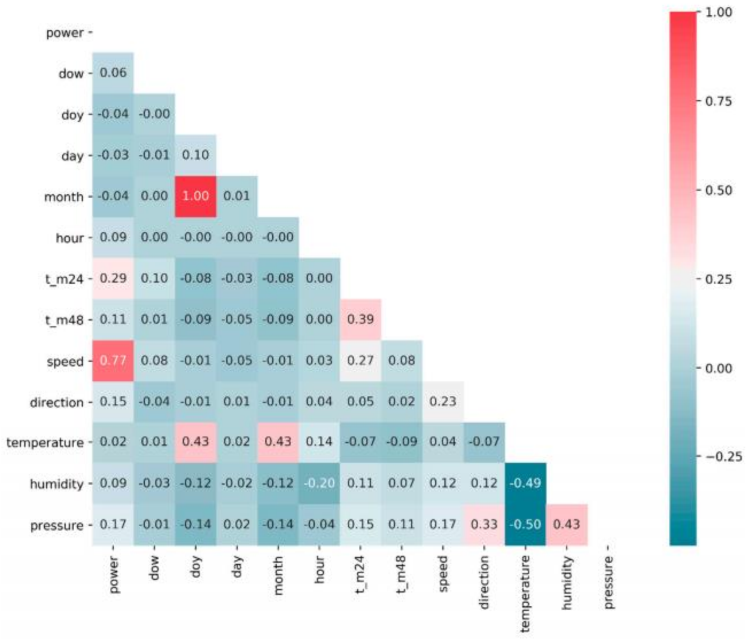

Figure 4.1: *Example of Pearson Correlation Heat Map (Zheng and Wu, 2019)*

In some cases, both data scientist and engineer do not necessarily comprehend the influence of each feature and are uncertain about which of the features that needs to be utilized in a ML project. In such a case, performing feature ranking is essential in order to examine the impact of each feature in terms of the desired output. This can be done using algorithms implemented in Python such as GB, RF and many more.

When performing feature selection, another important factor that needs to be considered is feature collinearity. Collinear features will provide a predictive model with almost identical information, which ultimately could lead to model confusion. Feature collinearity can be examined by plotting a Pearson correlation heat map for the input variables, as illustrated in Figure 4.1, and any collinear inputs should be dropped if present. However, if both inputs are of importance, the advised procedure would be to train two separate models for each collinear feature.

4.2.4 Data Scaling

To prevent learning algorithms form being biased to the magnitude of the data, input and output features must be scaled. Another benefit of data scaling is that the processing time of optimization algorithms used during model development can be sped up, as the input values will roughly be within the same range. However, it is worth noting that not all ML algorithms require scaling. A common rule of thumb is that algorithms which take advantage of similarities or remoteness between samples, such as ANNs, KNN, SVM and k-means clustering, are sensitive to feature transformations. On the other hand, some tree-based algorithms such as RF, GB, and decision trees (DT) are not impacted by feature scaling. This is due to the fact that tree-based models are not distance-based models and therefore capable of handling features which are varying in range. These types of models are often known as scale-invariant, as they do not require feature scaling in order to function. The following paragraphs will focus on the two most commonly used scaling methods, which are feature normalization and feature standardization.

4.2.4.1 Feature Normalization:

Feature normalization, which typically is referred to as min-max scaler, is utilized to assure that each feature will be scaled between 0 and 1. The method for normalizing a dataset can be seen in Eq (#), where *X'* is the normalized data point, *min(x)* is the minimum value of a parameter, and *max(x)* is the maximum value of a parameter in the data set. Feature normalization is a crucial step to prevent the model from being biased, which tends to happen when the existing features have different magnitudes. For example, suppose there are two features in a ML model. The first feature is the hookload, which can range between 50 000 - 350 000 pound-force. The second feature is the block position, which often ranges from 0 – 90 feet. If these features are left without feature normalization, it would result in a model which is bias towards the hook load because it has larger values. One of the main disadvantages of feature normalization is that smaller standard deviations is attained when the boundaries are reduced between 0 and 1, which in return could suppress the effects of outliers.

$$FeatureNormalization(X') = \frac{X - min(x)}{max(X) - min(x)} \tag{4.1}$$

4.2.4.2 Feature Standardization:

Feature standardization, or z-score normalization, is a method which transform each feature to a normal distribution with a mean of 0 and a standard deviation of 1. The method of standardization using a z-score is expressed in eq (#) as, where *X'* is the standardized data point, μ is the mean of a given parameter in the data set, and σ is the standard deviation of the same parameter in the data set. One should also note that standardization in fact does not transform the underlying architecture of the data. There are some issues with bias related to z-score standardization that needs to be accounted for as well. This can for instance happen when a learning algorithm such as Support Vector Machine (SVM) is utilized. SVM algorithms assume that the data presented are distributed around zero with an equal order of variance. If this perquisition is not met, the SVM algorithm will be biased towards features which has larger variance.

$$Z_{score}(X') = \frac{X - \mu}{\sigma} \tag{4.2}$$

Whether one uses normalization or standardization when scaling the data depends solely on the area of application. When working with clustering algorithms such as k-means clustering and hierarchical clustering, standardization is considered the appropriate way as these methods compares similarities residing with each feature based on span-measurements. For other algorithms such as ANN and image processing algorithms, normalization is a more fitting option.

4.2.5 Cross-Validation

Before applying a ML model of choice, implementation of s cross-validation method should be considered. This procedure is only necessary when a supervised ML algorithm is used and is not needed for unsupervised ML models, as they are only utilized for clustering. Details regarding supervised and unsupervised learning algorithms will be covered in chapter 4.3, as of now, just keep the abovementioned statement in mind. There are a couple of different methods to perform cross-validation, which will be discussed in the subsequent paragraphs.

4.2.5.1 Holdout Method:

The first and simplest method of cross-validation is the holdout method. This is basically a process where a small portion of the data is withheld or separated from the training data set. For an arbitrary dataset with given rows of data, the data is commonly split into two portions, whereas the portion which are used for training is larger than the one used for testing. In general, a 70/30 split is often used, where 70% of the data is used for training, while the remaining 30% is used for testing. A train-test split is chosen at random, and one should note that seed numbers can be defined in programming language such as Python to be able to replicate any process that relies on drawing random numbers. The largest downside to this method is the fact that training is done randomly, which makes this method incapable of defining which data-rows that are to be used for training and testing. This could in some cases be an inconvenience, and if the data set used for the analysis also happens to be small it could result in a very unexpected ML model outcome.

4.2.5.2 K-Fold Cross-Validation:

K-fold cross-validation is an extension of the previously mentioned holdout method. In this particular method, the given data set is split into a number of subsets K named folds, in which the holdout method is repeated K number of times. For a k-fold cross-validation, the data set will first be stocked arbitrarily. Afterwards, the slices will be applied to the train/test split. If we have a data set of 50 000 rows using K equal to 25, the k-fold cross-validation will be performed as follows:

1) The first 2000 rows of data will be used for testing, while the unused 48 000 rows are used for training
2) Next up, the rows for the testing set is shifted from row 0 – 2000 to row 2000 – 4000, while rows 0 – 2000 and rows 4000 – 50 000 is used for training.
3) This process is continuously repeated until K = 25 is reached.

Lastly, the average over 25 trials are conducted to acquire the evaluation metric. Evaluation metrics are tools used to measure the quality of a ML model. In ML, there are many different types of evaluation metrics available, and which type of metric that is used depends on the model type, algorithms and purpose of the model (DeepAI, 2020).

In short, the k-fold cross-validation uses all the data points as a test set one time and K-1 times for the training, which in this case would be 24 times.

One of the main advantages of the k-fold cross-validation is that it decreases the risk of high bias or variance (which will be covered in a later chapter), as most of the data set is being used for both training and testing. On the other hand, as K increases, the computation time also increases rapidly. Thus, one should perform an analysis to check if it is possible to reduce the number of K without lowering the model performance.

4.2.5.3 Stratified K-Fold Cross-Validation

The next method is the stratified k-fold cross-validation, which is commonly used when a large imbalance in the output variable of a ML model is present. Such an imbalance can often occur during classification problems. For instance, a dataset used for a spam filtering model could have 80% of its output classified as "Not spam", while the remaining 20% is classified as "Spam". As the number of outputs are split 80/20, we have a model imbalance. This is where the stratified k-fold cross-validation becomes convenient, as this cross-validation method assures that each fold is comprised of a relatively similar sample percentage of each target class. For instance, if K = 5 for an arbitrarily number of rows, each of the five folds will be split 80/20 with data that are classified as "Not Spam" and "Spam". The stratified k-fold cross validation is an alternative to the k-fold cross-validation and is only required when there is a model imbalance in the data set.

4.2.5.4 Leave-P-Out Cross-Validation:

The last method is the leave-P-out cross-validation (LPOCV). In this process, a set of P datapoints are excluded of the training data. If a set of N datapoints are available, then N-P datapoints are used for training the ML model, while P points are used for testing. This process is repeated up until all possible outcomes are divided and tested. Subsequently, the average of all outcomes is calculated to attain the evaluation metric. For example, if N=10 rows of data and P=2, 8 rows will be used for training and 2 rows will be used for testing. This step is repeated until all combinations are divided and tested. LPOCV can produce very accurate projection of the evaluation metric. However, the LPOCV method is computationally demanding for large datasets, which is why the leave-one-out cross-validation (LOOCV) is more commonly used. In LOOCV, P=1 which ensures that the number of folds is equal to the amount of data points. Thus, using LOOCV is recommended over LPOCV where P is greater than 1. Overall, this particular approach is only recommended to be used when working with smaller data sets.

4.2.5.5 Blind Set Validation

In addition to splitting the data into training and test sets, a blind set validation must also be applied in order to evaluate a ML models precision in terms of its predictive capabilities. A blind set is a portion of the data that is withheld from the beginning, which is neither used for testing nor training. It is worth mentioning that many research papers, as well as other writings, use blind sets and testing interchangeably and should not be confused with the test set used for cross-validation.

Utilization of a blind set allows a comprehensive and unbiased assessment of a model's accuracy. A blind set is typically in the range of 10% - 20% of the original data and is set aside prior to any ML analysis. The remaining percentage of the data is split into training and testing and are fed to the model. When training of the ML model is finished, the trained model is exposed to the blind set to examine the model's performance on unseen data. The outcome will provide the examiner with metrics regarding the accuracy for the -training, -test and -blind set.

4.2.6 Bias – Variance trade-off

A key concept in ML is what many refers to as a bias-variance trade-off. Bias is defined as a ML models inability to recognize the true relation between the data input and output (Belyadi and Haghighat, 2021). A ML model with high bias will oversimplify the problem at hand, consequently leading to poor model performance. Such an incidence could for instance occur when a model uses a linear regression method for a non-linear dataset.

Variance is defined as the variability of a models prediction capabilities for a given data-point (Belyadi and Haghighat, 2021). A high variance is an indication that the model is able to memorize its training data remarkably well. However, memorization in ML is not considered as a good trait, because a model with high variance will perform poorly when it comes to predictions on unseen data. This is due to the fact that the model's proficiency in terms of generalization becomes insufficient, which will affect the model performance negatively.

The ratio of bias and variance present in a model will impact its complexity. If a ML model is too simple, it will be labeled as underfit. An underfit model indicates that the model has a low variance and a high bias, which makes it unable to capture most of the underlying complexity in the training dataset, as seen in Figure 4.2. If the opposite holds true, the model will possess a high variance and low bias. Thus, it will be labeled as overfit. Creating a model which is neither underfit or overfit is the ideal solution, which would require a low variance and low bias.

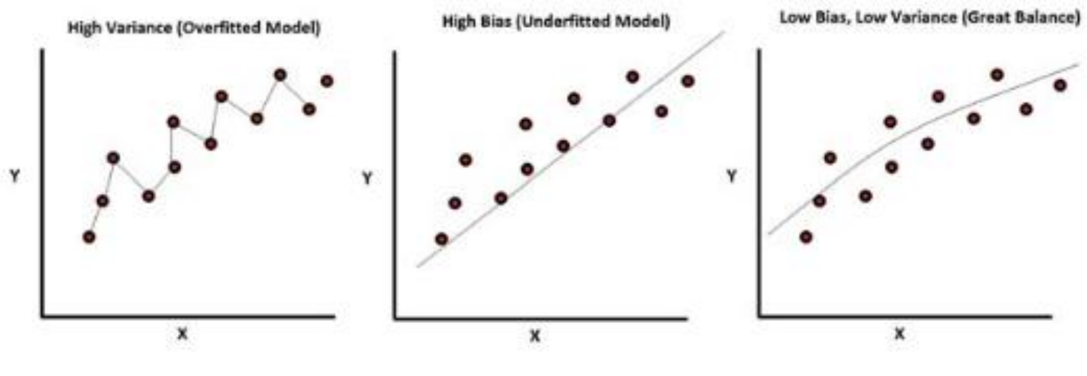

Figure 4.2: *Overfitted, underfitted and balanced model (Belyadi and Haghighat, 2021).*

4.2.7 Model Implementation.

After completing the abovementioned procedures, the last step would be to select an appropriate algorithm to address the challenge at hand. For this phase, a variety of algorithms should be evaluated in order to decide which of the algorithms that are deemed most effective. In a practical setting, there is no such thing as an all-purpose algorithm. Therefore, the factor that dictates the most suitable algorithm for a given problem would be the underlying data which has been examined in the preprocessing stage. There are many different types of ML models with a variety of algorithms that are available, and the categories which they are commonly divided into will be discussed in the next chapter. After model implementation and successful training, the ML model are eligible to be imported and actualized into a real-time data management system.

4.3 Machine Learning Types:

Approaches to machine learning are often split into two main categories which are based on two different traits. The first group are based on the models learning methodology, which can be divided into supervised and unsupervised learning. It is also worth mentioning that there are extensions of these categories, where the methods functionalities can be considered as a hybrid between that of the supervised and unsupervised learning method. The second grouping is based on the ML models field of application, which broadly can be separated into classification or regression methods. ML is comprised of a variety of methods, but all methods are either supervised or unsupervised, and either regression or classification (Bangert, 2021). Any ML problem can be split into these categories. Thus, understanding how these groups behave are of essence, and will therefore be discussed in this chapter.

4.3.1 Supervised and Unsupervised Learning:

Supervised learning refers to empirical data in which a set of inputs (features), and target outputs (labels) are available for the model. On the other hand, unsupervised learning refers to datasets where the only information that is offered to the model are the input features. Thus, the purpose of unsupervised learning becomes to seek out existing patterns in the input data (Nketah, 2016). Some examples of unsupervised ML algorithms are k-means clustering and principal component analysis (PCA). In the O&G industry, these type of algorithms can be used for purposes such as lithology classification and liquid-loading detection (Ansari et al., 2018, Singh et al., 2020). Examples of algorithms that fall underneath the category of supervised learning are ANNs, multi-linear regression (MLR), logistic regression (LR) and SVM (Belyadi and Haghighat, 2021). Supervised learning methods are in the O&G industry utilized for objectives such as prediction of multiphase flow regimes and productivity forecasting in reservoirs (Alhashem, 2019, Han et al., 2020).

Figure 4.3 has been added to attain a better understanding of how the two different methods work and demonstrates two situations where a ML model is supposed to learn the difference between a collection of datapoints. By looking at the first example illustrated to the left, one can observe that there exists both triangles and circles in the dataset, and that these points are clearly different. Hence, we have a supervised problem, and the ML model needs to be taught and understand where to split the datapoints in order to accurately separate the figures. On the right side we have an unsupervised problem, as the ML model are only handed a collection of datapoints depicted as circles. For such a case, the ML model must be able to learn that the points can be logically divided into clusters, where the points residing in each cluster are greatly alike while being dissimilar to the points residing in other existing clusters. The clustering is of course only valid for a given measure of similarity which would make sense in the situation related to the problem one wishes to solve.

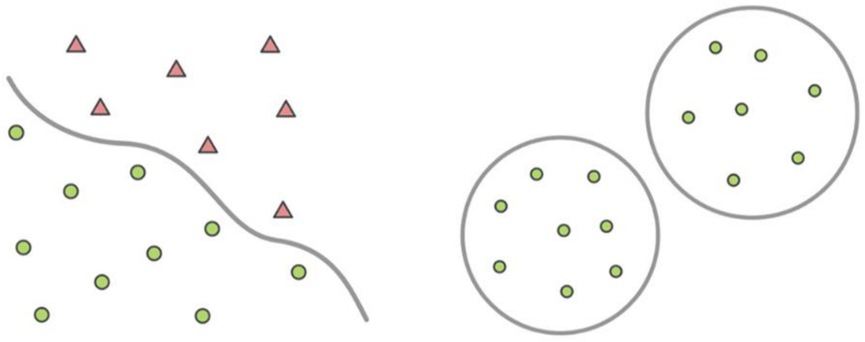

Figure 4.3: *Difference between supervised learning (left) and unsupervised learning (right) (Bangert, 2021)*

The major difference between supervised and unsupervised methods is that for a supervised model, the predictions which are provided are always comparable with the pre-existing labels. This is not the case for unsupervised models, as there are no existing labels which the predictions can be compared against (Belyadi and Haghighat, 2021). This particular difference would imply that supervised methods would be better suited to reproduce a precise outcome. However, the option of being able to compare our predicted outputs with the true label variable might lead to issues related to overfitting.

4.3.2 Classification and Regression Methods

For the second grouping, the approach of classification has the intent of placing a datapoint into its respecting category. By contrast, regression methods has a goal of being capable of computing a numerical value based on the task that is at hand. Figure 4.4 displays the difference between these two methods. The first example to the left illustrates a classification problem, as the goal for the ML model is to differentiate one category from the other, namely triangles from circles. The example to the right shows a typical regression problem, as the aim of the ML model is to reproduce a set of consecutive, numerical values that are able to capture the generic trend of the dataset.

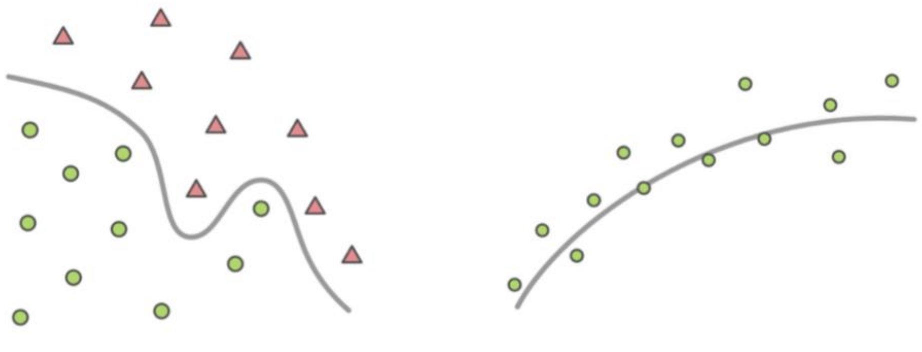

Figure 4.4: *Difference between the classification method (left), and the regression method (right) (Bangert, 2021)*

4.4 Artificial Neural Networks

Artificial neural networks, or neural networks, are another subset of ML and is often described as a network of neurons which mimics the human brain. In reality, ANNs are simply a mathematically modelled tool. Neural networks are usually made up of a series of neurons that are ordered in a sequential layered structure, where each input and output variable can be designated to a node. These nodes are then accumulated into layers and structured in a specific way. First, we have an input layer, which is followed by one or more sets of hidden layers that connects the input and output layer. Lastly, all connected nodes from the input layer to the last hidden layer is then connected to an output layer as seen in 4.5. The number of hidden layers, as well as the number of nodes in each of the hidden layers that connects the inputs with the outputs are defined as the topology of a neural network, and is determined by the model architect (Bangert, 2021).

The strength of the connection between a node from one layer to another are named weight, and is denoted as w_{ij}, where i and j corresponds to the number of nodes in the input and output layers (Belyadi and Haghighat, 2021). The value designated to a weight represent the importance of a node-connection with respect to the desired output. A positive weight number indicates that a node-connection has a positive influence on the output, while a negative weight value indicates that a node-connection has negative influence on the output.

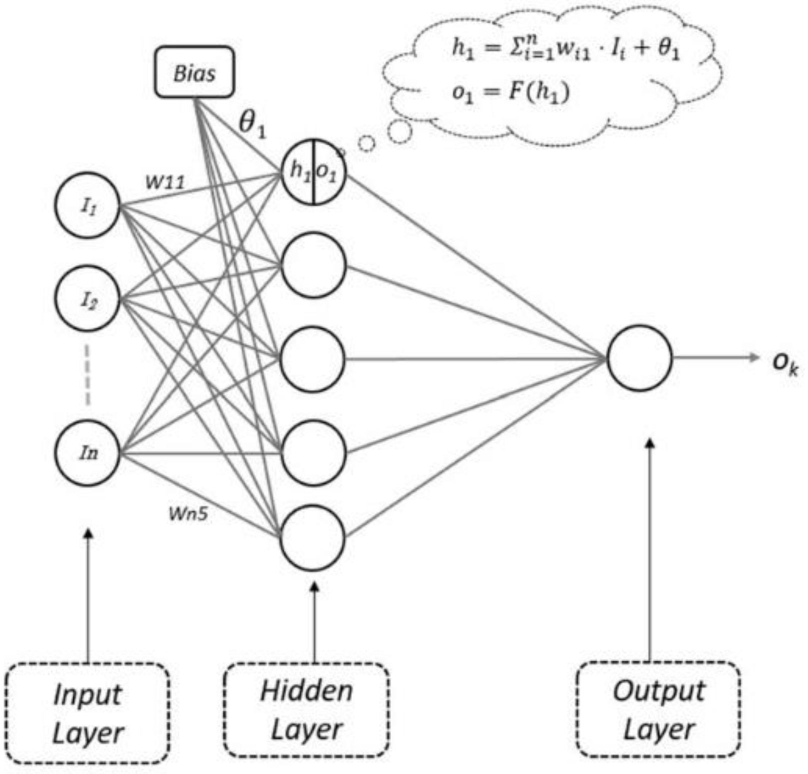

Figure 4.5: *Neural network schematic with input layer, output layer and one hidden layer (Belyadi and Haghighat, 2021)*

For a neural network each node in a hidden layer picks up signals from the input layers and accounts for biases. To clarify how the procedure in figure 4.5 works, assume that an arbitrary number of “n” input nodes exist. Now, the action potential of a node *j* in a hidden layer h_j, can be calculated according to equation 4.3 (Belyadi and Haghighat, 2021):

$$h_j = \sum_{i=1}^{N} w_{ij} * I_i + \theta_j \tag{4.3}$$

Where I_i is a signal from the ith input node, w_{ij} is the signal weight from node *i* to node *j*, and θ_j is the bias. The bias in this equation is represented as a pseudo-node, which has a standard output value of 1 and is utilized if the input value is 0.In order to solve the system of equations for all the nodes in a ANN, equations 4.3 can be rewritten in a matrix form in the following way (Belyadi and Haghighat, 2021):

$$H = W * I + B \tag{4.4}$$

Where H is the action potential vector, W is the weights matrix, I is the input vector and B is the bias vector.

4.5 Activation Functions

After the action potential is calculated, the next step in the process is to calculate the value of an output node o_j. This value is calculated using an activation function, denoted as *F*, which uses the action potential of node h_j to determine the amplitude of the output signal o_j. Based on a bias value and the algorithm that is taken advantage of, the activation function generates a value between -1 and 1, or 0 and 1 (Belyadi and Haghighat, 2021). These functions are utilized in the hidden layers of an ANN to capture non-linear relations which resides in the model.

In general, there are three activation functions which are commonly used in neural networks, namely: sigmoid, hyperbolic tangent (TanH), and the rectified linear (ReLU) activation function (Belyadi and Haghighat, 2021, Bangert, 2021, Google, 2021, Browniee, 2021). In older literature, the sigmoid and TanH activation functions were the preferred methods of approach (Kızrak, 2019).

However, throughout the recent years, the ReLU activation function has become the most used functions for hidden layers (Belyadi and Haghighat, 2021, Browniee, 2021, Bangert, 2021). This is due to the fact that this particular function is considered simple to implement, but also effective at overcoming limitations found in the Sigmoid and TanH activation functions.

4.5.1 ReLU Activation Function:

The ReLU activation function works in the following way. If the action potential h_j is less than 0, the activation function would output the value of 0. For any other value of the action potential, which is larger than 0, the output value of the activation function is equal to h_j. This relation is simply a maximum function and can be described in the two following ways (Bangert, 2021, Belyadi and Haghighat, 2021):

$$ReLU(h_j) = Max(0, h_j) \tag{4.5}$$

$$o_j = ReLU(h_j) \begin{cases} 0, & h_j > 0 \\ h_j, & h_j \geq 0 \end{cases} \tag{4.6}$$

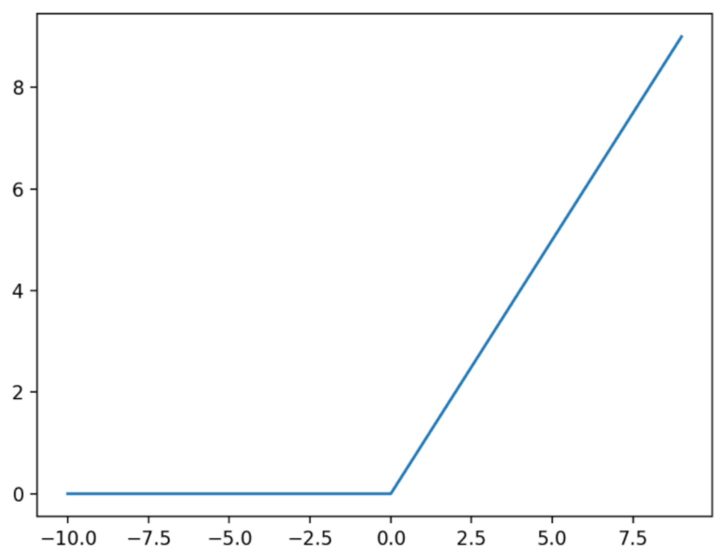

Figure 4.6: *Plot of inputs vs outputs for the ReLU Activation Function (Browniee, 2021)*

4.5.2 Sigmoid Activation Function:

The sigmoid activation function, or otherwise known as the logistic function, takes any real value as an input and then outputs a value in the range between 0 and 1. As the input values increase (becomes more positive), the output value will approach 1. On the other hand, if the input value is decreasing (becomes more negative), then the output value will approach 0 (Browniee, 2021). The sigmoid activation function is calculated in the following way (Belyadi and Haghighat, 2021):

$$o_j = Sigmoid(h_j) = \frac{1}{1 + e^{-h_j}} \tag{4.7}$$

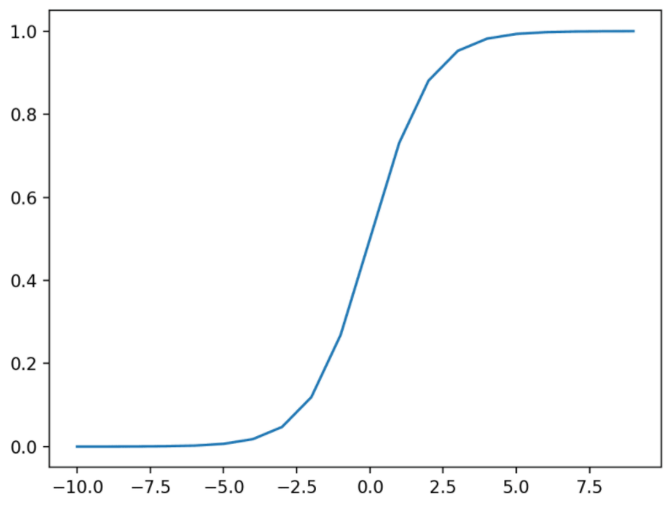

Figure 4.7: *Plot of input vs outputs for the Sigmoid Activation Function (Browniee, 2021)*

4.5.3 TanH Activation Function:

The hyperbolic tangent activation function is similar to the sigmoid activation function. It only differs from the sigmoid function in terms of output values, which range from -1 to 1. For TanH, as the input values increase, the output values will move towards 1. When the input values decrease, the output values will move towards -1. The TanH activation function is calculated in the following way (Belyadi and Haghighat, 2021):

$$o_j = TanH(h_j) = \frac{1 - e^{-2h_j}}{1 + e^{-2h_j}} \tag{4.8}$$

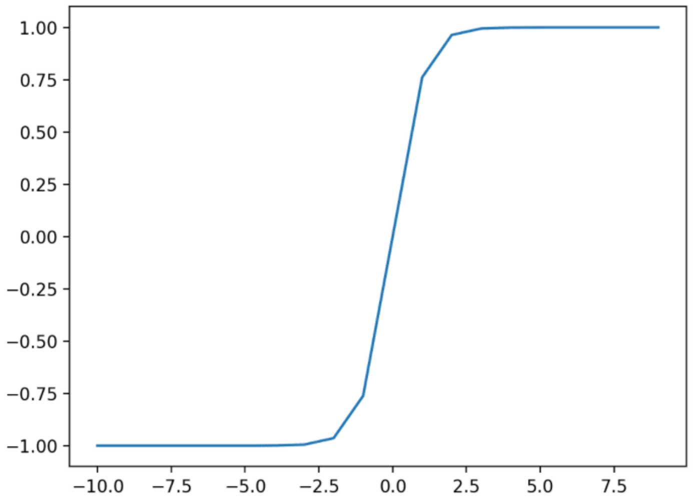

Figure 4.8: *Plot of inputs vs outputs for the TanH Activation Function (Browniee, 2021)*

Similar to the action potential, calculating the output vector, *O*, by can done be applying an activation function to all of the elements in vector H in this particular way (Belyadi and Haghighat, 2021):

$$O = F(H) \tag{4.9}$$

This process can be repeated for each hidden layer as long as the output of the first hidden layer is the input for second hidden layer, the output of second hidden layer is the input for the third and so on, until the final hidden layer is reached. This procedure is termed feedforward, when there is no feedback from the output results back into the hidden layers and nodes (Bangert, 2021).

4.5.4 Backpropagation Technique:

The objective of any training procedure in neural networks is to optimize the calculated wrights until the model is capable of estimating the output values properly. The common method used achieve this is the backpropagation technique, which requires that the weights are being updated during the training process in order to minimize the model error. These errors are essentially the difference between the target values and the actual network output. The backpropagation technique is composed of a feedforward calculation to get the initial output, an error calculation that is propagated back into the network, and a weight adjustment phase which is based on the magnitude of errors. When a full cycle of backpropagation is completed, it is called an epoch. Epochs are used to decide when to end the training procedure. In practice, one need to run through more than one epoch to achieve satisfactory model accuracy, which can be adjusted when tuning the model.

There are many neural network optimization algorithms available that can utilized to reduce the error between the predicted output values and the true values (Zhang et al., 2020). Although they have fundamental differences, the backpropagation method for the last layer is similar for most of them. In the last layer, weight adjustments based on error for δ*k*, the error gradient, at each neuron k of the output layer, is calculated through the following equation (Belyadi and Haghighat, 2021):

$$\frac{\partial e_k}{\partial w_k} = \delta_k = (t_k - o_k)\frac{\partial F(O_k)}{\partial w_k} \tag{4.10}$$

where o_k and t_k are the predicted output and target values for neuron *k* in the output layer and *F* is the activation function, while w_k is the weight of the neuron *k*. If the target value is not available in the hidden layer, the weight optimization parameter can be estimated as (Belyadi and Haghighat, 2021):

$$\delta_j = \frac{\partial F(O_j)}{\partial W_j}\sum_{k=1}^{n} w_j \delta_k \tag{4.11}$$

The new weights (W^n_{jk}) can then be reshaped into Equation 4.11 (Belyadi and Haghighat, 2021):

$$W_{jk}^{n} = W_{jk}^{n-1} + \Delta W_{jk}, \quad \Delta W_{jk} = \alpha * \delta_k \tag{4.12}$$

where α is labelled as the learning rate, which can take a value between 0 and 1. The value set for the learning rate decides how fast the model should learn, as well as how quickly the weights should be adjusted. If α is set to a minor value the learning rate will be fairly slow, while a large value of α can cause instabilities in the neural network. Another issue related to a high value of α could be that the model gets stuck in a local optimum and thus be unable to locate the true optima. A parameter μ, defined as the momentum, is commonly used to solve this problem. The momentum ensures that the step-size towards the global minimum increases rapidly before slowing down again, so that the model can avoid getting stuck in a local minimum. The weight can therefore be calculated as follows (Belyadi and Haghighat, 2021):

$$\Delta_{j,\ k}^{n} = \alpha * \delta_k + \mu * \Delta W_{j,\ k}^{n-1} \tag{4.13}$$

It is important to mention that when one starts the training process of a neural network, random initial values should be assigned to the weights. These values can be randomly selected by the model architect, and can range between -1 and 1, which then are assembled as a uniform probability distribution (Belyadi and Haghighat, 2021, Bangert, 2021).

4.5.5 Model Loss:

For each epoch that is run in a neural network, the goal is to minimize the output value of a given loss function. In other words, the model developer wishes to reduce the global error residing in the neural network. There exist a variety of loss functions for neural networks. However, implementation of the appropriate function depends on what type of problem the developer is facing, and each function should be considered accordingly (Browniee, 2019). Nevertheless, the overall concept remains the same. As the number of epochs increases, the error of training and test data sets is expected to reduce. It is recommended to keep training the model if the error residing in the test data set keeps on decreasing.

The training process should be halted once the error of the test set starts to increase while the error of the training set keeps on decreasing, as this indicates that the neural network is beginning to remember the training pattern. As previously mentioned in chapter 4.2.7, model memorization is something that should be avoided, because it will have a negative impact on the model's generalization capabilities. Lastly, when the epoch of which the smallest error of the test data set is attained, the model can be saved and utilized on unseen data. This concludes the chapter on neural networks. Before moving on, it is important to mention that the workflow provided in chapter 4.2 should be used alongside the information given in this chapter to accurately train a neural network model

4.5.6 Hyperparameters:

Hyperparameters are values that must be chosen manually for any network model. These values are utilized to attain the most optimal model during training and are often chosen through trial and error. These values include, but are not limited to:

- Batch Size
- Learning Algorithm
- Learning algorithm parameters such as momentum and learning rate
- L1 and L2 regularization
- Number of hidden layers and neurons therein
- Activation functions

Hyperparameter selection is an important process and will directly affect the convergence of a neural network.

The number of hidden layers and neurons needed in a network differs from task to task. As of today, there are no specific rules for selecting the right numbers of hidden layers in a neural network, nor to select the number of neurons in each layer. However, certain guidelines can be followed to effectively reduce the time spent on setting up the model architecture (Sachdev, 2020):

- If the data is linearly separable then no hidden layers are necessary.
- If the data is less complex and have fewer dimensions or features, then 1 to 2 hidden layers would suffice
- If the available data have large dimensions or features, receiving an optimal solution would require 3 to 5 hidden layers

It is worth mentioning that increasing the number of hidden layers could contribute to solve real world problems, but one should keep in mind that having an excessive number of layers might potentially lead to overfitting of the model.

Deciding the correct number of neurons in the hidden layers is also of importance. Too few, and the network will be unable to learn. Too many will exponentially increase the network training time and may also lead to overfitting. As for choosing the number of hidden layers, there exist a heuristic set of rules to choose the number of neurons in hidden layers. These are as follows (Sachdev, 2020):

- The number of hidden neurons should be between the size of the input layer and the output layer
- For each subsequent layer, the number of hidden neurons should keep on decreasing to get closer to pattern and feature identification of the target class.

In addition, the optimal number of hidden neurons in a hidden layer, *k*, can be approximated in a number of ways (Heaton, 2008, Azoff, 1993, Gençay and Qi, 2001), but the following approach by Freisleben (1992) is the one most commonly used.

$$k = \sqrt{mn} \tag{4.14}$$

where *n* is the number of neurons in the input layer and *m* is the number of neurons in the output layer. The usual approach in terms selecting the appropriate number of hidden layers is to increase the number of nodes during training until the networks starts overfitting. Then, one corrects the number of nodes to de

5 Parameters Affecting the Hookload

Recognizing how the hookload behave during drilling and tripping, as well as for changing wellbore geometries is of essence when working with the differential sticking model. Hence, this chapter will describe the parameters that most significantly affects the hookload in order to gain a deeper physical understanding surrounding this topic.

5.1 Definition

The hookload is a measurement of all forces acting on the drill string, that are pulling down on the hook (Schlumberger, 2020). The total force includes the weight of the drillstring and any additional equipment (i.e. the topdrive) that is suspended on the drilling hook, as well as any forces that tend to reduce this weight, such as mechanical and hydraulic frictional forces, possible restrictions, etc.

The hookload value is attained as a function of the deadline tension and the lines between the crown and travelling block (Luke and Juvkam-Wold, 1993):

$$HKL = F_{dl}n \tag{5.1}$$

where F_{dl} is the deadline tension, and n is the number of sheaves between the crown block and travelling block. In reality, there are other factors such as friction between sheaves, effects of tension applied by umbilicals, etc, which will also affect the hookload measurements (Cayeux et al., 2015). However, these only amount to small deviations in actual readings and Equation 5.1 will therefore hold true for explanatory purposes.

5.2 Buoyancy:

The weight of the drillstring which is recorded at the deadline is dependent on the density of the drill string and drilling fluid in the borehole. An example of the forces acting on a drillstring submerged in drilling fluid for a vertical well is illustrated in Figure 5.1. In such a case, there are two forces which affects the hookload, the buoyant and gravitational force.

Buoyancy is the upward force caused by a fluid which opposes the downward movement of an object. When working with drillstrings, the drilling fluid is what opposes the weight of the drillstring as it tries to move downwards.

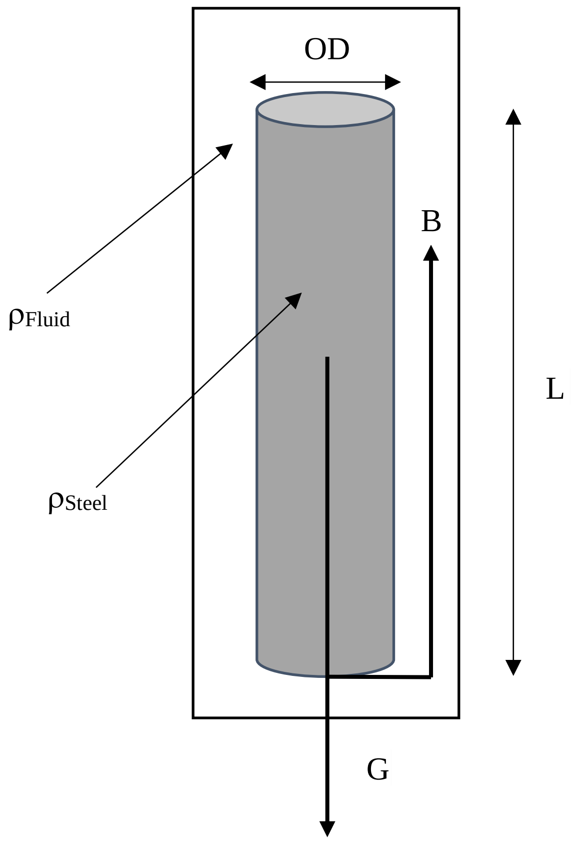

Figure 5.1: *An illustration of the forces acting on a single drillstring submerged drilling fluid for a vertical well with no restrictions.*

According to Archimedes principles, the buoyancy of an object is equal to the weight of the displaced fluid in which it floats (Aadnoy and Kaarstad, 2006). The net weight of a drillstring component is calculated in the following way (Bourgoyne et al., 1986):

$$W = \rho_{steel} A \beta g L \tag{5.2}$$

where W is the buoyed weight of a single drillstring component suspended in fluid, A is the cross sectional area of the pipe, L is the length of the drillstring component, g is the gravitational force and β is the buoyancy factor, which can be expressed as:

$$\beta = \left(1 - \frac{\rho_{mud}}{\rho_{steel}}\right) \tag{5.3}$$

For this example, the simplest form of calculating the buoyancy factor is used. Thus, equation 5.3 is only valid if both the inside and outside of the string is submerged in the same fluid. For cases where a string is composed of varying pipe sizes, or the fluid property on the inside and outside differs, equation 5.3 needs to be altered to take these factors into account (Aadnoy and Kaarstad, 2006).

When the pipe is pulled through the submerged fluid, the velocity of the drill string, along with the viscosity of the fluid will also affect the hookload. This physical phenomenon is called fluid drag and will be discussed in further detail later in this subchapter.

Friction

Friction is defined as the force of a solid or liquid surface that resist the relative motion of an object (Britannica, 2021). The resistance caused by the relative horizontal motion between the surface of two solids is called dry friction and is commonly used to describe the frictional force present in the wellbore between the drillstring and the wellbore wall.

Friction:

The value of the friction force can only be determined empirically. As of today, dry friction is mostly calculated using the Coulomb friction model, which defines the frictional force as:

$$F_f \leq \mu F_N \tag{5.4}$$

where F_f is the force of Coulomb friction, μ is the coefficient of friction and F_N is the normal force. Depending on the force that an object is pulled with, F_f may take any value from zero up to μF_N. In addition, F_f always works in the opposite direction of which the moving object is displaced. When the drillstring is hoisted, the frictional force is exerted downwards, and when the dirllstring is lowered, the frictional force is exerted upwards. Figure 5.2 is a simplified illustration of the forces which acts on the drillstring in as it moves downwards.

F_N is the force which acts perpendicular to the contact force exerted by an object onto a surface, in order to prevent the two objects from penetrating each other. For this argument to hold true, the normal force must be equal to the gravitational component which works in the opposite direction of the normal force. Hence, the normal force can be expressed as:

$$F_N = mg \sin \alpha \tag{5.5}$$

We can then rewrite Equation 5.4 to the following form:

$$F_f \leq \mu mg \sin \alpha \tag{5.6}$$

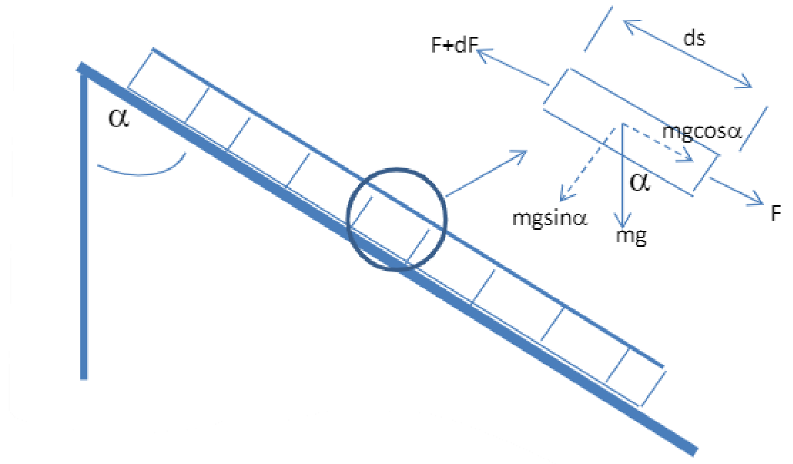

Figure 5.2: *Free body diagram of a mass element on an inclined plane depicted as a drillstring, where (F + dF) is the axial force pulling on the drillstring.*

Friction forces will occur at any instance where the drillstring is in contact with the wellbore walls. Theoretically, this means that dry friction will be absent in vertical wells. In reality, there always exist some extent of contact between the walls and drillstring in vertical sections. In addition, one can tell from Equation (5.6) that the normal force is expected to increase as the angle of inclination increases. This phenomenon will especially affect horizontal wells, as they consist of hold, build-up and drop sections, which causes larger parts of the drill string to rest against the borehole wall.

5.3 Fluid Friction:

Fluidic drag friction, or hydrodynamic viscous force, is a friction force which is exerted onto a submerged object by the surrounding fluid relative to the direction of the moving object. In essence, it follows the same principles as dry friction. For a drillstring submerged in drilling fluid, depending on the direction its moving (ie, RIH or POOH), the hydrodynamic viscous force will work in the opposite direction. This force is caused by shear stress along the drillpipe wall hindering the displacement of the drilling fluid. Following Newtons law of viscosity, the shear stress can be defined as (Bourgoyne et al., 1986):

$$\tau_w = \mu \frac{du}{dy} \tag{5.7}$$

where *u* is the fluid velocity and *y* is the distance of the drillpipe from the wall. Hence, *y* can be replaced with the difference in radius between the wellbore wall and the pipe, measured from the pipe center, as illustrated in figure 5.3. In addition, the wall shear stress works in the opposite direction of the pipe movement, which allows Equation 5.7 to be rewritten as:

$$\tau_w = -\mu \frac{du}{dr} \tag{5.8}$$

The inhibiting force due to wall shear stress can also be written as:

$$\tau_w = \frac{F_D}{A_p} \tag{5.9}$$

where A_p is the lateral surface area of the pipe wall. If one assumes a constant fluid velocity in the annulus, du/dr = 0, and F_D, can be expressed as:

$$F_D = -\frac{2\pi r L\mu u}{(2R-2r)} = -\frac{\pi r L \mu u}{(R-r)} \tag{5.10}$$

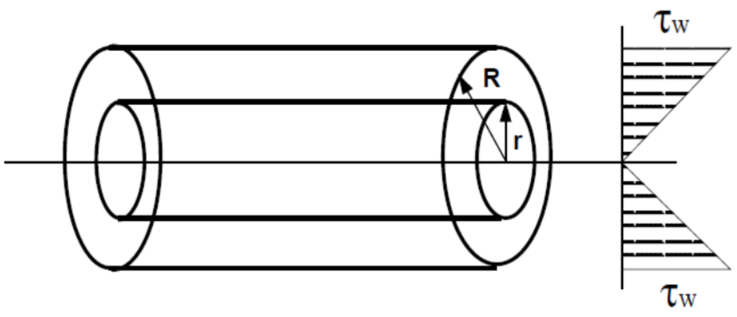

Figure 5.3: *Wall Shear Stress Profile of a pipe*

The hydrodynamic viscous force will increase quickly as the clearance between wellbore and pipe decreases. Assuming a constant velocity profile in the annulus is a simplification. In reality, the velocity profile will behave parabolic, as seen in Figure 5.4

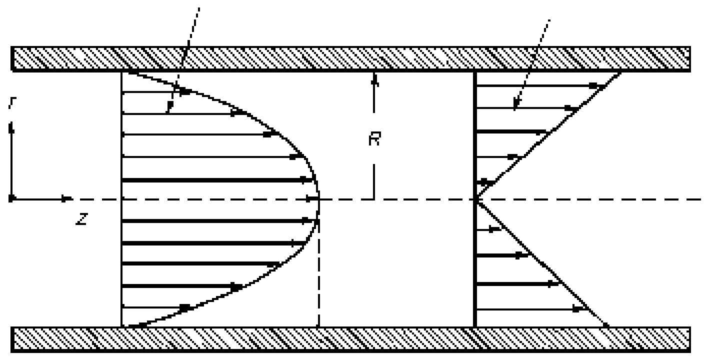

Figure 5.4: *Velocity profile (left) and shear stress profile (right) in a fully developed laminar flow in a pipe (William, 2020)*

From 5.10 one might get the impression that the fluidic drag force is impacted by the velocity and viscosity equally. However, the viscosity of the fluid will in fact impact the fluid friction force to a greater extent than the velocity (Polak and Lasheen, 2001).

As the angle of inclination increases, the effects of fluid drag will be of less importance. For wells with an inclination larger than 60°, the contribution of this force in terms of the overall wellbore friction factor is minimal. This is due to the fact that sliding friction will start to dominate all other effects (Maidla and Wojtanowicz, 1987).

5.4 The Coulomb Friction Relationship

μ, the coefficient of friction, is a dimensionless scalar value which is defined as the ratio of the force required to move two surfaces across each other and the normal force acting between those two surfaces (Shrivastava, 2018). Dry friction can be divided into two parts, namely:

- Static Friction (μ_s)
- Kinetic Friction (μ_k)

Static friction is the value of μ_s when there is no relative motion between two surfaces, while kinetic friction is the value of μ_k when the opposite holds true. The applied force onto a rigid object must overcome the static friction to initiate motion. As the applied force increases, so will the static friction. Once the maximum value of the static friction, $\mu_s N$, is surpassed, the object will start to move, and the applied force required to keep the object in motion will drop to its maximum kinetic value, $\mu_k N$ (Byrom, 2007). This relationship can be seen in figure 5.5, indicating that the applied force required to initiate motion of a rigid object is larger than the force required to uphold its motion.

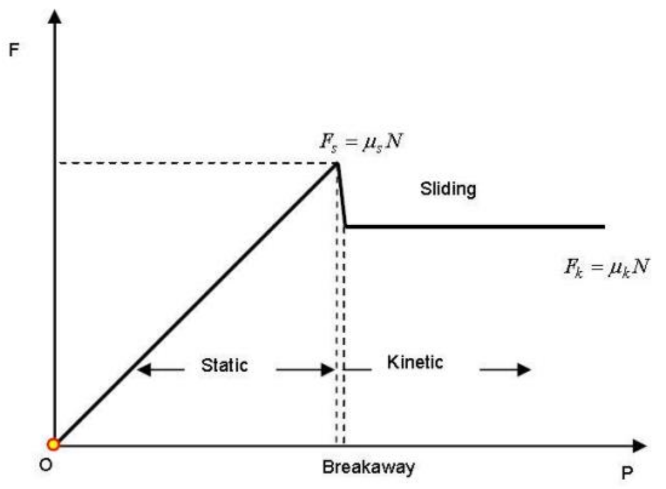

Figure 5.5: *Ideal behavior of Coulomb friction for a rigid body. The static friction will increase until motion is initiated, then changes to a kinetic friction force (Tveitdal, 2011).*

Therefore, the relationship between the static and kinetic friction can be defined as:

$$\mu_k < \mu_s \tag{5.11}$$

As a stand is being tripped, the whole drillstring transitions from standing still, to moving. Because of this, one can expect the hookload to behave somewhat similar to the Coulomb friction model during hoisting or lowering operations. The only exception is that the required force to initiate motion will increase with each stand, as more weight is added to the string.

However, a drillstring is more comparable to an extensible body rather than a rigid one. Therefore, we can think of the drillstring as a body divided into pieces with a weightless spring connected between each piece. The total weight of the string is still the same, but the weight will now be distributed equally between each piece. For the sake of this example, suppose that four pieces of pipe are present, as illustrated in figure 5.6. As force is gradually applied, the only piece that the force will act on is the one to the left. No force will be transmitted to the next piece before the first block starts to move. When static friction in the first block is overcome, the applied force will drop back to its kinetic value.

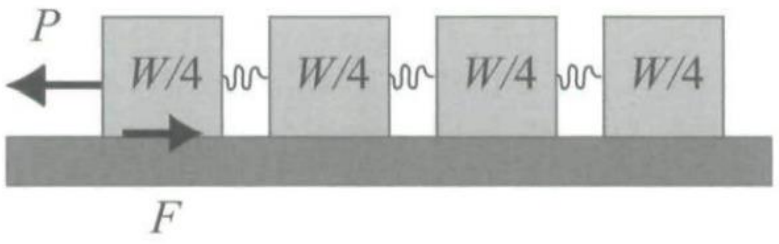

Figure 5.6: *An extensible body model, where P is the applied force, F is the friction force, and W is the weight of the block (Byrom, 2007)*

As one continues to increase the applied force, the same sequence will take place in each piece until the final piece is initiating motion. This effect will ensure that the total difference between the maximum static friction force and the maximum kinetic force will only be one fourth of what it would have been for a single, rigid body, as illustrated in figure 5.7 (Byrom, 2007).

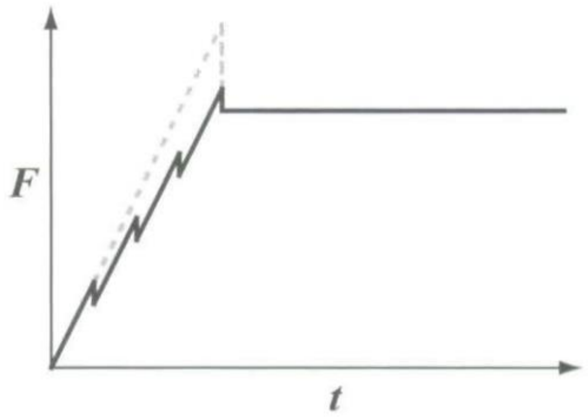

Figure 5.7*: Friction force in an extensible body (Byrom, 2007)*

If one applies this logic for a long drillstring consisting of *n* number of pipes, the friction force at all times can be expressed as the sum of drillpipe segments in motion multiplied with the kinetic friction coefficient, plus the weight of a single stationary drillpipe segment multiplied with the static friction coefficient (Byrom, 2007):

$$\mu_k < \mu_s \rightarrow F = \mu_k \frac{W}{n}(n-i) + \mu_s \frac{W}{n} \tag{5.12}$$

where *n-i* is the number of pipes in motion, and *W* is the weight of the pipe. Now, suppose that an infinitely long well is drilled. The limit of the friction force as the number of pipes goes to infinity can be written as:

$$\lim_{n\to\infty} F = \mu_k W \lim_{n\to\infty}\left(\frac{n-i}{n}\right) + \mu_s \lim_{n\to\infty}\left(\frac{W}{n}\right)$$

$$\lim_{n\to\infty} F = \mu_k W(1) + \mu_s(0)$$

$$\lim_{n\to\infty} F = \mu_k W \tag{5.13}$$

Equation 5.13 has not been added to prove that a static friction coefficient is inexistent, but to highlight that for longer well-sections, real-time hookload data might display a different behavior during tripping than what we would expect based on an analytical model. Looking at Figure 5.6 and Equation 5.13 again, another possibility may perhaps be that the height difference between μ_s and μ_k could become small enough to go unnoticed for longer drillstrings. Although the reasoning surrounding an extensible body might be over simplified, it shows that for an extensible body, there exists some degree of deficiency in terms of the Coulomb friction model, which should be taken into account during the development of the ML model.

5.5 Effects of Bore-Hole Friction with changing inclination:

The mechanisms that can lead to increased friction in the borehole covers a wide range, including the factors mentioned up until this point in chapter 5.1, chapter 3, and more. When calculating the effects of friction in a wellbore, the common approach is to estimate an overall friction factor which approximately considers all possible mechanisms working against the pipe movement.

Due to the various effects of friction present in a wellbore, the hookload value is prone to be affected by the geometry of a drilled well. As of today, the impact of borehole-friction can best be observed through an extensively used method in the O&G industry called torque and drag (T&D) modeling. Drag is an overall term that refers to all forces that restricts the relative motion of a drillpipe and should not be confused with fluidic drag. There have been suggested a variety of models to describe the effects of borehole friction, but most models base their concepts off the one presented by Johancsik et al. (1984).

For calculations related to torque and drag, a single pipe element filled with fluid is considered. Johancsik states that the forces acting on this element are the axial tension, friction force, buoyed weight, and the normal force. In addition, Johancsik also states that the resisting forces depend on the weight of the string which is supported by the wellbore. The axial tension of a pipe element can be expressed as (Johancsik et al., 1984):

$$F_2 = F_1 + W\cos\theta \pm \mu F_N$$

$$F_2 = F_1 + W\cos\theta \pm \mu\sqrt{(F_1\Delta\alpha\sin\theta)^2 + (F_1\Delta\theta + W\sin\theta)^2} \tag{5.14}$$

where F_2 and F_1 is the axial tension at the top and bottom of the element, W is the buoyed weight of the pipe element, and $\Delta\alpha$ and $\Delta\theta$ is the change in wellbore azimuth and wellbore inclination, respectively. F_1 depends on the operation that takes place. For tripping in, out or when the bit is rotated off bottom, F_1 is equal to zero, and is given a positive value during drilling to simulate the WOB. In addition, the friction force is positive for tripping in, and negative for tripping out. For any straight section with no curvature, $\Delta\alpha = \Delta\theta = 0$. Thus, for a pure vertical section Equation 5.13 is reduced to:

$$F_2 = F_1 + W \cos\theta \tag{5.15}$$

and for inclined well sections, Equation 5.13 becomes:

$$F_2 = F_1 + W\,(\cos\theta \pm \mu \sin\theta) \tag{5.16}$$

Lastly, for a horizontal well section, Equation 5.13 can be rewritten as:

$$F_2 = F_1 \pm \mu W \sin\theta \tag{5.17}$$

From Equation 5.14, 5.15 and 5.16, one can tell that the effects which governs the axial force of the drillstring changes as the well transitions from vertical to horizontal. For the vertical section, the axial tension is mostly dominated by the normal force due to gravity, while in the horizontal section, axial tension becomes a product of the wellbore friction.

To further investigate the impact of wellbore geometry on the hookload, a discretized T&D model based on Equation 5.13 – 5.16 was developed in Python, and its source code can be found in Appendix A. It takes basis in a fictive well with a 900m vertical section, 900m build-up section and a 900m horizontal section. The differential Equation used in this model are integrated over 30m long segments to replicate the typical length of a standpipe. Inclination in the build section increases in steps of 3 degrees every 30 meter, until an angle of 90 meter is reached. Lastly, F_1 is set to 0 and change in azimuth is considered fixed and neglected.

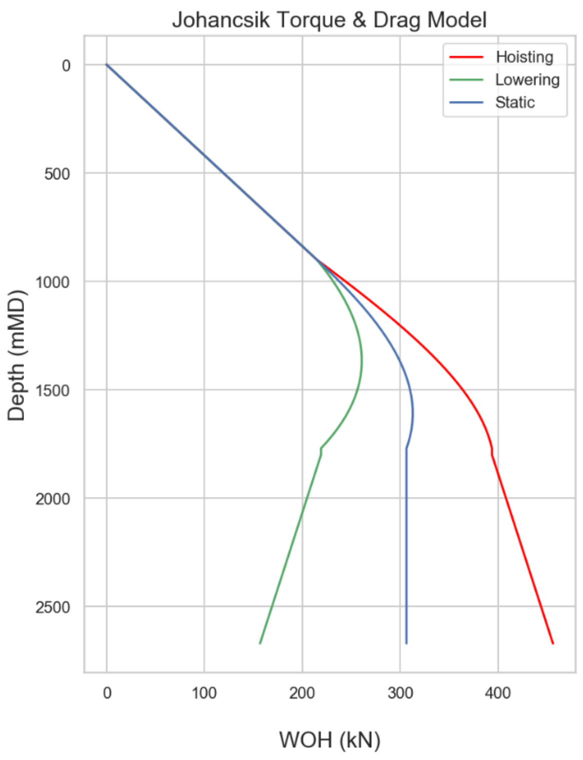

Figure 5.8: *T&D plot based on the model by presented by (Johancsik et al., 1984). The HKL keeps decreasing with depth as the pipe is lowered and keeps increasing with depth for hoisting operations.*

Figure 5.6 displays the T&D model output, where the hookload value is plotted against the measured depth of the well. For all three lines, the hookload keeps steadily increasing until the build-up section is reached. This is because Johancsik’s model assumes no friction in a purely vertical well. Thus, the hookload is only increasing due to the weight of each consecutive stand that is added onto the drillpipe.

From the build-up section and onwards, the graphs start to deviate from each other. For tripping in, one may observe that the hookload keeps decreasing as the pipe is lowered into the hole. This is because the frictional force works in the opposite direction relative to the pipe movement and will therefore contribute to alleviate the weight experienced on the drilling hook. For tripping out the opposite holds true, as the frictional force resists the upward movement of the pipe and causes an increase in the experienced weight on the drilling hook. Based on the hookload observations made from the T&D model, one can expect the real-time HKL data to show a similar decreasing trend for RIH procedures, and an increasing trend for POOH procedures.

6 Experimental Setup

The following subchapters will go further into detail describing the typical architecture of the DS model used by Exebenus, as well as how data was gathered, cleaned, and extracted for training purposes, followed by the programmed setup for the ML model.

6.1 Model Type:

The DS model developed by Exebenus is considered as a univariate time-series model. This type of model is simply a neural network which only uses a single parameter, such as the HKL, to make predictions on time-series data. Such a model has the added benefit of avoiding issues related to feature collinearity, and their DS model falls under the category of supervised regression methods, where the model type implemented is a recurrent neural network (RNN). RNNs are a subclass of ANNs, which are specifically designed to handle sequential data, and is the model that is most often considered when working with real-time surface data in the O&G industry (Bangert, 2021).

As opposed to ANNs, which have connections that transitions strictly from left to right, RNNs have incorporated connections that creates cycles, as illustrated in Figure 6.1. These cycles act as memory for the model to recall previous data, which makes a RNN capable of using output values at former time steps to estimate the output at the current time step (Belyadi and Haghighat, 2021, Bangert, 2021). There exist a variety of RNNs and they mainly differ in how the cycles are mathematically represented in the neural network (Bangert, 2021). The type of RNN which Exebenus uses is called long short-term memory (LSTM). It is a model developed by Hochreiter and Schmidhuber (1997), and is currently the most frequently used recurrent network model.

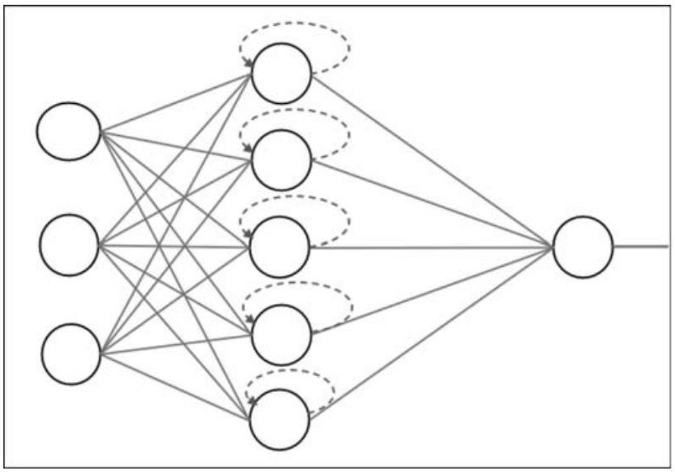

Figure 6.1: *Illustration of a recurrent neural network (Belyadi and Haghighat, 2021)*

The architecture of a LSTM network is made up of linked memory cells that regulates the flow of information during training when sequential data is presented to the model. Each memory cell includes three operating units called an input gate, forget gate and output gate, which interact with each other to retain the error through sequential backpropagation (Hochreiter and Schmidhuber, 1997). The purpose of these gates is to control whether data should arrive, be kept, or leave the cells. An example schematic of a single memory cell and its components can be seen in Figure 6.2.

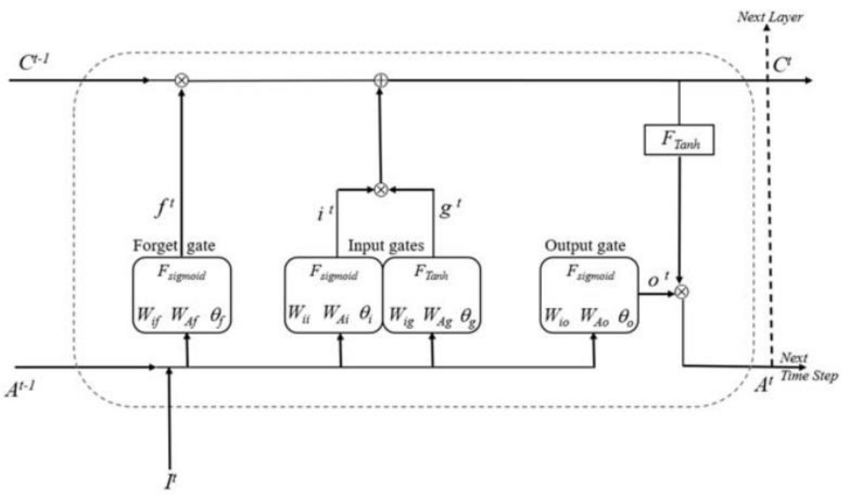

Figure 6.2: *Schematic of a LSTM cell and its corresponding cell-units (Raschka and Mirjalili, 2019)*

One important feature of a LSTM cell is the cell state, *C*, which can be treated as the cell's memory. The cell state is responsible for preserving or forgetting the former information the model has received, as it moves through all cells present in the network. For each time step, the previous cell state, C^{t-1}, the previous hidden layer state, A^{t-1}, and the input data at the current time step, I^t, enter the cell. As seen in Figure 6.2, I^t and A^{t-1} initially enter the forget gate. Then the output at the forget gate, f^t, determines the information that should be kept or forgot from the past time step, and can be calculated through Equation 6.1 (Raschka and Mirjalili, 2019):

$$f^t = Sigmoid(W_{if}I^t + W_{Af}A^t + \theta_f \tag{6.1}$$

where W_{if} and W_{Af} is the weight matrix in the forget gate for I^t and the current hidden layer state, A^t respectively, and θf is the bias in the forget gate.

After the f^t is calculated, A^{t-1} and I^t moves through the input gate, which updates the cell state, *C*, through this set of following equations (Raschka and Mirjalili, 2019):

$$i^t = Sigmoid(W_{ii}I^t + W_{Ai}A^t + \theta_i \tag{6.2}$$

$$g^t = TanH(W_{ig}I^t + W_{Ag}A^t + \theta_g \tag{6.3}$$

$$C^t = (C^{t-1} * f^t) + (i^t * g^t) \tag{6.4}$$

where W_{ii}, W_{Ai}, W_{ig}, and W_{Ag} are the weight matrices in the two "boxes" of the input gate for I^t and A^t, while θ_i and θ_g are the biases for each box.

Lastly, the output gate updates A^t based on the examining the revised cell state, C^t, through the following equations (Raschka and Mirjalili, 2019):

$$O^t = Sigmoid(W_{iO}I^t + W_{AO}A^{t-1} + \theta_O \tag{6.5}$$

$$A^t = O^t * TanH(C^t) \tag{6.6}$$

where W_{iO} and W_{AO} are the weight matrix in the output gate for I^t and A^{t-1} respectively, while θ_O is the bias of the output gate.

6.2 Data collection and cleaning

Data cleaning and preprocessing was done utilizing the Pandas library in Python. It is a well-built library which is designed to make working with time-series and structed data both easy and intuitive, and is able to handle CSV, Excel and other type of file formats (McKinney, 2010).

For this research project, Exebenus initially provided surface drilling data from a total of 13 wells. Well A to E were drilling data for vertical or slightly deviated wells and were therefore discarded. The remaining wells F, G, H, I, J, L, M included drilling data for horizontal wells with a sampling frequency of ten seconds. After further inspection of the dataset for these well it was decided that well G, I, J, L, M and O had to be discarded due to either of the following reasons:

1) Presence of large intervals where logging sensors have failed to record data
2) Presence of sensor drifts which were difficult to correct
3) Anomalies in the drilling data which could not be interpreted, nor were informed about in the daily drilling reports.

Thus, only Well H and F were considered feasible for training purposes. Well H were then selected as it was the dataset which contained the least amount of errors. The dataset spans over the course of three weeks and consist of 162060 rows and 135 columns for different surface parameters. Furthermore, the dataset contains all drilling activities initiated at the rig. In other words, both drilling, circulation, and various tripping operations are logged and present for all variables.

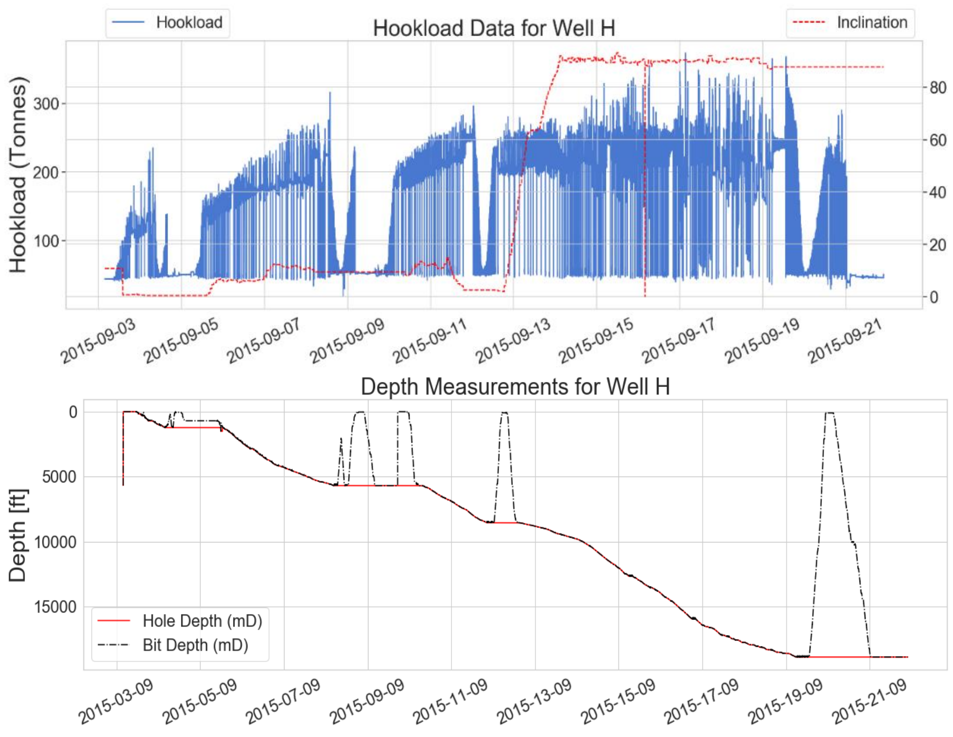

Figure 6.3*: Raw Data for HKL, Bit Depth and Hole Depth*

Since Exebenus already have established the inputs necessary for their DS-model, only the HKL, hole depth, and bit depth were cleaned for feature extraction purposes. In addition, inclination has also been added alongside the HKL to investigate possible changes that might occur as the well transitions from vertical to horizontal. Other parameters, such as the position of the travelling block, were only used to examine anomalies in the abovementioned variables.

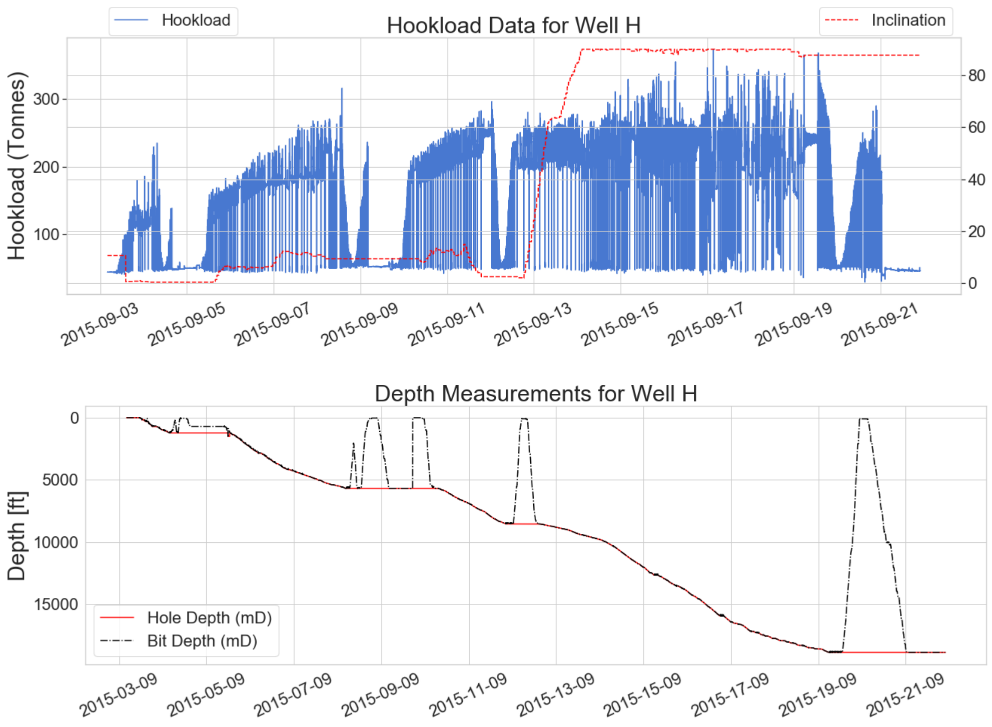

Figure 6.4: *Cleaned Data for HKL, Bit Depth and Hole Depth*

Figure 6.3 displays the raw data of the four parameters. By visualizing the raw data, it becomes clear that there are some outliers present which must be investigated. For the measured depths, inspection showed that both hole depth and bit depth were at 5657 ft for the first ten minutes of the dataset, before making a sudden change to 0 ft. In addition, all other parameters present during this interval were also constant. Because of this, the first 10 minutes of the dataset was removed. For the HKL, some of the bottom values (during connection) shows sign of sudden drops. Inspection of neighboring HKL values to the outliers conclude that these drops were just faulty data. Thus, they were replaced with the average value of the neighboring points. The outlier present in the inclination data is simply a situation where the sensor has temporarily stopped. The sudden drop present in the plot were therefore replaced with a value of 90, as the data indicates a drilling scenario through the horizontal section. The cleaned version of the raw data is shown in Figure 6.4.

6.3 Model Features and Feature Extraction:

Through their research, Exebenus discovered that prior to a differential sticking event, the static friction during tripping displays an increasing trend (Meor Hashim et al., 2021a). An increase in friction will lead to the overall drag force increasing. When a pipe or casing is RIH, this phenomenon can be observed as a decrease in the local minimum HKL value, as more force is required to overcome the static friction. Figure 6.5 illustrates such as case, where three stands of casing is being tripped in.

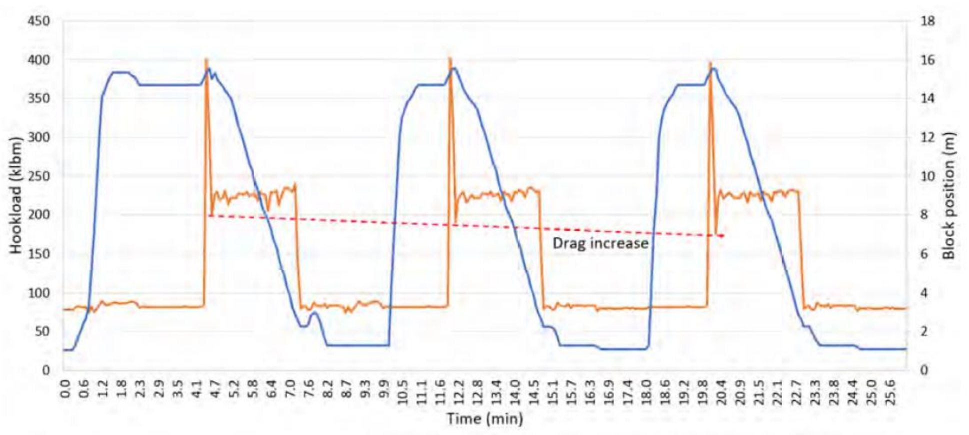

Figure 6.5*: Sign of decrease in the minimum HKL values as a consequence of increased drag (Meor Hashim et al., 2021a)*

When tripping out, drag force will work in the opposite direction and will therefore pull the pipe further downwards. Thus, for a POOH operation, an increasing trend in the maximum HKL values will indicate a possible DS situation. During drilling, the ML model will monitor the trend of the maximum HKL, or more commonly known as the pick-up weight, after each connection. Hence, the ML model by Exebenus only predicts the trend of these specific HKL points, and not the whole HKL pattern. These points are then fed to a set of algorithms that provides real-time warnings based on the calculated probability of getting stuck. This approach, as opposed to other ML model which often use multiple input parameters (Abbas et al., 2019, Salminen et al., 2017), have provided Exebenus with a simple yet robust model. The following subchapter will focus on how these points were extracted from the utilized dataset.

6.4 Feature Extraction

The first step to extract the relevant features would in theory be to do feature ranking and selection. This process was skipped since the features already have been defined through the company. Thus, the next logical step was to differentiate between the different operations in the dataset. The operations were divided into three main categories, namely drilling, tripping in, and tripping out. In order to accomplish this, the simplest method was to check the position of the bit. If the bit position is equal to the hole depth, a drilling operation is taking place and if the bit is off bottom, a tripping operation is ensuing. This reasoning was used to establish the following set of conditions:

$$\begin{gathered} HKL\ Tripping\ In = \\ (Hole\ Depth > Bit\ Position)\ \&\ (Bit\ Position\ Next > Bit\ Position) \end{gathered} \tag{6.7}$$

$$\begin{gathered} HKL\ Tripping\ Out = \\ (Hole\ Depth > Bit\ Position)\ \&\ (Bit\ Position\ Next < Bit\ Position) \end{gathered} \tag{6.8}$$

$$\begin{gathered} HKL\ Drilling = \\ (Drilling\ < \ Tripping\ In)\ \&\ (Tripping\ In \leq Drilling\ Next) \end{gathered} \tag{6.9}$$

where *Bit Position Next* is a condition which checks the bit position for a set interval ahead in time. By comparing the future value with the current value, one can determine whether the bit is moving towards or away from the bottom of the wellbore, which is deemed necessary to decide if the pipe is hoisted or lowered. *Drilling Next* serves the same purpose as *Bit Position Next*, as it checks if future values of the bit position are equal to the hole depth. This condition was added to handle situations where for instance vibrations in the drillstring could cause the bit to temporarily bounce off the bottom of the wellbore, making the set conditions recognize the operation as tripping and not drilling.

As mentioned in previous chapters, additional stands of drillpipe are added to the drillstring as a drilling operation ensues. Such an action will according to the initially set conditions also fall under the category of tripping in, as the future position of the bit will be further down in the well than the current position, after a stand has been added. Thus, equation 5.7 needs to be changed into the following form to work around this issue:

$$\begin{gathered} HKL\ Tripping\ In = \\ (Hole\ Depth > Bit\ Position)\ \&\ (Bit\ Position\ Next > Bit\ Position) \\ \&\ (HKL\ Drilling\ \neq HKL\ Tripping\ In) \end{gathered} \tag{6.10}$$

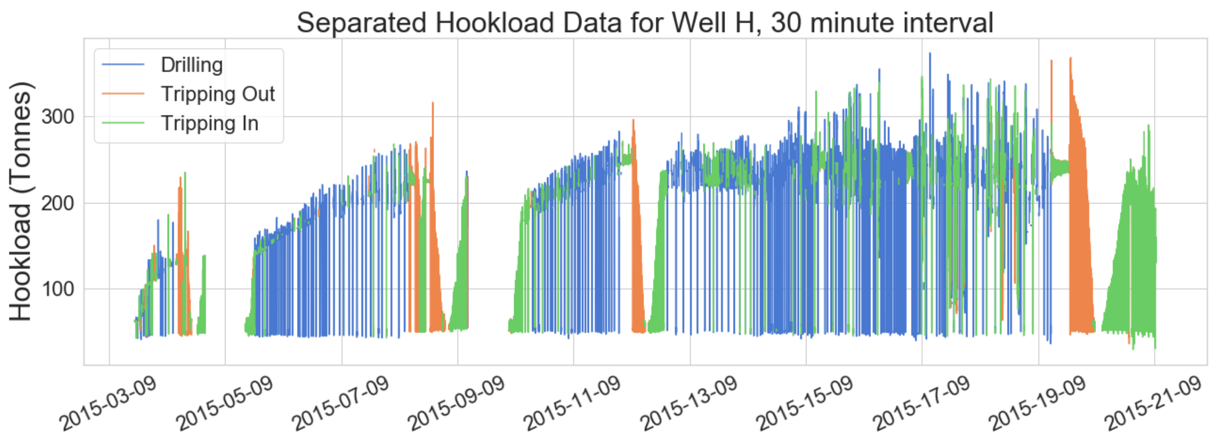

Figure 6.6: *HKL data separated into Drilling, Tripping In, and Tripping Out*

The most suitable timestep ahead of time needed to compare against the current position of the bit was determined using a trial and error method. It was concluded that comparing the current bit position against the bit position 180 rows ahead of time (30 minutes), yielded the most accurate results. Figure 6.6 shows the separated HKL for drilling, tripping in, and tripping out. Even with the supplementary condition added to equation 6.10, there are still some areas between 13th of September and 19th of September which still is categorized into the wrong group. How the issue was handled will be discussed further down in this section.

After the data was separated into its rightful categories, the next step was to extract the input values according to the model developed by Exebenus. This was done using the "findpeaks" function in Python, which allows the user to locate all peaks and valleys in a dataset. These values were then stored in new groups corresponding to each of the three operations. The findpeaks function is rather convenient, as peaks and valleys can be located above or below, as well as between a set of given points. Nevertheless, the function will pick up all maximum and minimum values within its given criteria, which also includes unwanted peaks and valleys caused by noise in the dataset. Therefore, a function was created to extract the true values necessary from all three groups. For drilling and tripping out, the function first detects a minimum value which corresponds to the HKL reading during connection. Then the function searches for the next maximum value above a given threshold. For tripping in, the first two steps are similar to the previous groups. The only difference is an added condition that makes the function search for the first minimum value after the slips has been removed. After all relevant points have been detected, the algorithm adds them to a new column in the dataset which can later be used for training of the ML model.

The algorithm was able to extract a total of 1692 points that could be utilized for training. However, when checking the new column, it came to light that many of points related to tripping in were wrongly extracted according to the expected output of the algorithm. Figure 6.7 shows a sample of the HKL data from Well H for 12 stands that has been tripped in, where the red dots are the extracted HKL features. Through closer inspection of this data, it was discovered that the distinct minimum HKL value which appears after the slips have been removed is absent for many of the stands. The reason why these distinct points disappear was unclear. The two reasons which were found plausible were that either the sampling frequency of the dataset was too low, or that the friction coefficient transition from static to dynamic with a small enough margin to go undetected, which was discussed as a possibility in chapter 5.4.

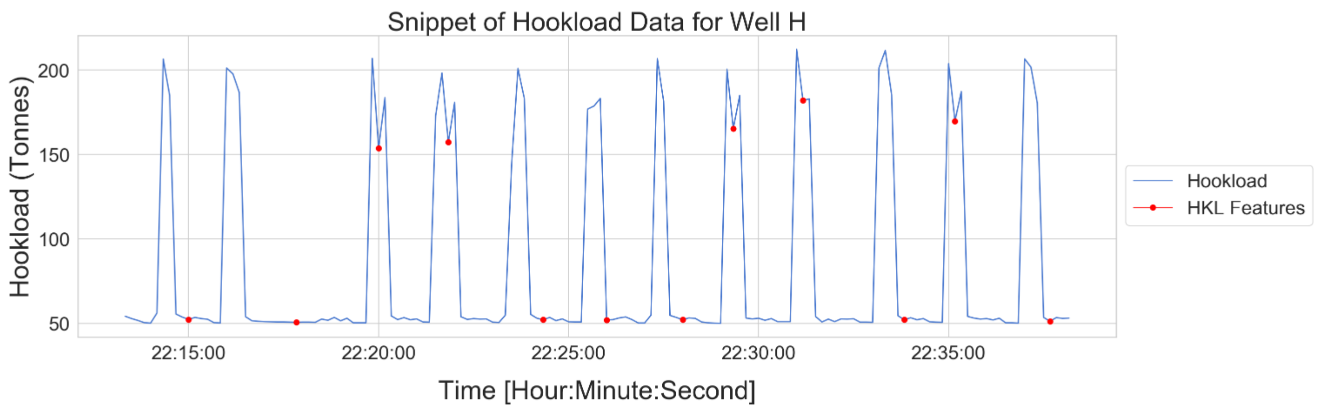

Figure 6.7: *HKL data with wrongly extracted feature values*

Therefore, discarding Well H and using another dataset with a higher sampling frequency was considered an option, as this would clarify the issue. A new set of wells, P, Q, R, S and T were received for this purpose, with a sampling frequency of 5 seconds and inclination up to 80 degrees. Inspection of the newly received data indicated that well R, P and T suffered from the same problems as those mentioned in chapter 6.2 and were therefore discarded. In addition, Well S only contained data for three stands being tripped and could not be used for model training. Lastly, well Q were missing the hole depth which were used as an input parameter for the feature extraction algorithm. Since time were limited when the problem in Figure 6.7 came to light, well Q were also discarded.

A Machine learning model trained with wrong inputs will ultimately lead to poor model performance due to consequential errors. Hence, the best solution which did not require a new dataset was to remove all faultily extracted points. These points displayed similar behavior, in the form of being registered as a HKL value during connection. The HKL during connection is expected to remain approximately the same throughout the entire operation. Thus, all faulty feature values below 60 tonnes were removed. During this process, most of the faulty extracted points due to wrong categorization were coincidentally removed. After cleaning the features, the amount of inputs available to the model were reduced from 1692 to 1177 datapoints.

Figure 6.8 displays the same sample as Figure 6.7 after removal of faulty points. As one may notice, and indirect consequence of using the method of removal is that the model no longer receives an input for each consecutive stand that is being tripped.

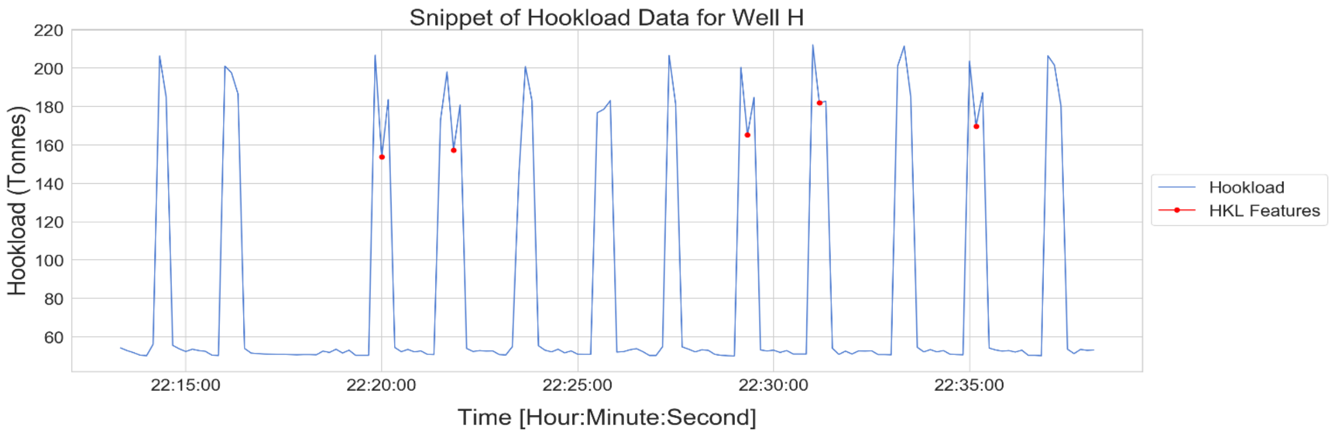

Figure 6.8: *HKL data after removal of faulty extracted points*

Although having faulty points for features is of bigger concern in terms of model training, LSTM models are designed to store a set amount of past points to predict the next sequence of values. In addition, LSTM models are also able to readjust based on the new information it receives. By removing the faulty points, the time and distance between points used for acquiring, predicting, and updating the model is increased, and could possibly impact both model training and testing.

6.5 Model Setup and Training

The LSTM model was developed using Tensorflow, a robust Python library which has the capability of easily creating deep learning models (Abadi et al., 2016). In addition, a high-level neural network API called Keras (Chollet, 2015), is run on top of the Tensorflow library package to fully advance the ML model. For predictions, the LSTM was set up to predict the next 4 points in the dataset based on information from the last 10 points The whole architecture of the model can be found in Appendix B.

After feature extraction was conducted, the relevant HKL features were exported to a new data frame. Data partitioning was done using a 70/30 split, where the first 70% of the data was used for training and the last 30% for testing. From the training set, 20% were used for validation, and the data was scaled using the min-max method, as described in chapter 4.2.4. Thus, a total of 855 points were used for training and validation, while the remaining 367 points were used for testing, where the training set consisted of hookload data for both the vertical and horizontal section. Model training were conducted using a trial and error method, and the final hyperparameters tuning which achieved the lowest loss can be seen in Table 6-1.

To evaluate model loss, the quality of the hookload predictions was analyzed in terms of the Mean Absolute Error (MAE) and residual distribution. The value of the MAE can be calculated in the following way (Belyadi and Haghighat, 2021):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{6.11}$$

where n is the total number of observations, y_i is the true value of the ith output and $\hat{y}_I$ is the predicted model value of the ith output. Utilizing MAE as opposed to other loss functions were found to be most appropriate approach when the issues observed during feature extraction was discovered. This is because the method of MAE is more robust in terms of feature outliers and are therefore better equipped work around the longer intervals where the relevant features are missing.

Table 6.1: *Final Hyperparameter Tuning for the LSTM model*

Hyperparameters	**Experimental Range**	**Choice**
Batch Size:	12-100	64
Activation Function:	ReLU	ReLU
Optimizer:	Adam	Adam
Epochs	50-500	150
Learning Rate:	0-1	0.01
L1 regularization:	0.1, 0.001, 0.0001, 0.00001	0.001
L2 regularization:	0.2, 0,002, 0.0002, 0.00002	0.002
Number of Hidden Layers:	4	4
Number of neurons in each hidden layer:	128, 64, 32, 16	128, 64, 32, 16
Dropout:	0.8 – 0.1	0.3, 0.2, 0, 0

The attained reduction in training and validation loss using the MAE can be seen in Figure 6.9. For the training set, the minimum loss achieved were equal to 0.0456, while the minimum loss for the validation set were equal to 0.024. A lower validation loss compared to training loss can in many cases be a sign of overfitting. However, regularizes and dropouts are hyperparameters that the ML models only apply during training.

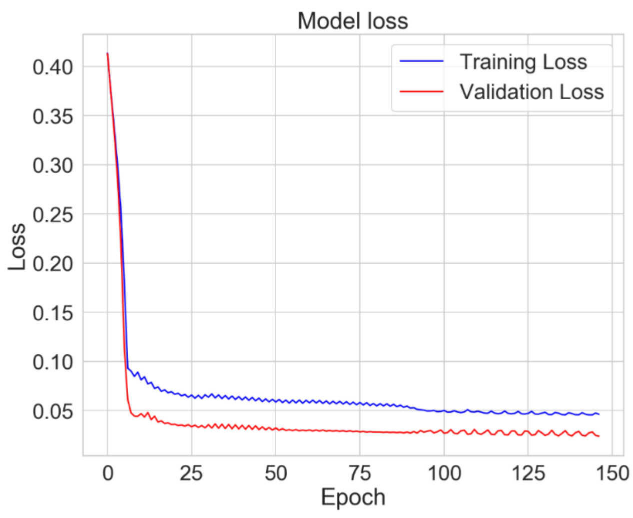

Figure 6.9: *Attained Model loss for the training and validation set.*

The outcome is a lower accuracy during training, in exchange for better generalization capabilities to unforeseen data. Therefore, the validation loss was expected to be lower than the training loss. The plotted loss in Figure 6.9 can also be seen to slightly fluctuate, which indicates that the regression model did not stabilize at the global minimum. Nevertheless, the fluctuations are small, and the model were therefore considered sufficient for testing

7 Results and Discussion

This chapter will discuss the results obtained from developed LSTM model, as well as its overall performance in terms of the evaluation metrices. In addition, reflections will be made upon the observed changes in the horizontal section, as well as challenges surrounding the scope of the research.

7.1 Model Prediction and Perfomance Evaluation

Figure 7.1 shows the original values and the forecasted output for the test set of Well H. The predicted period spans over the course of three days and includes operations for both hoisting and lowering. The dates displayed on the x-axis corresponds to the last interval of the drilling operation, which confirms that the drillstring is being POOH and the production casing is being RIH. In accordance with the T&D model provided in chapter 5.3, one may also observe that the HKL values for tripping out is larger than for tripping in, even though the casing weighs more than the drillpipe. The higher HKL values due to friction as each stand is hoisted creates a steeper slope which needs to be predicted, but the ML model appears to have adapted accordingly.

By looking at Figure 7.1, it seems that the developed model has performed well on the unseen data it was presented with, but due to large amounts of clustering it is difficult to determine if the statement is correct. The features and predicted values were plotted separately to study the output better, as seen in Figure 7.2.

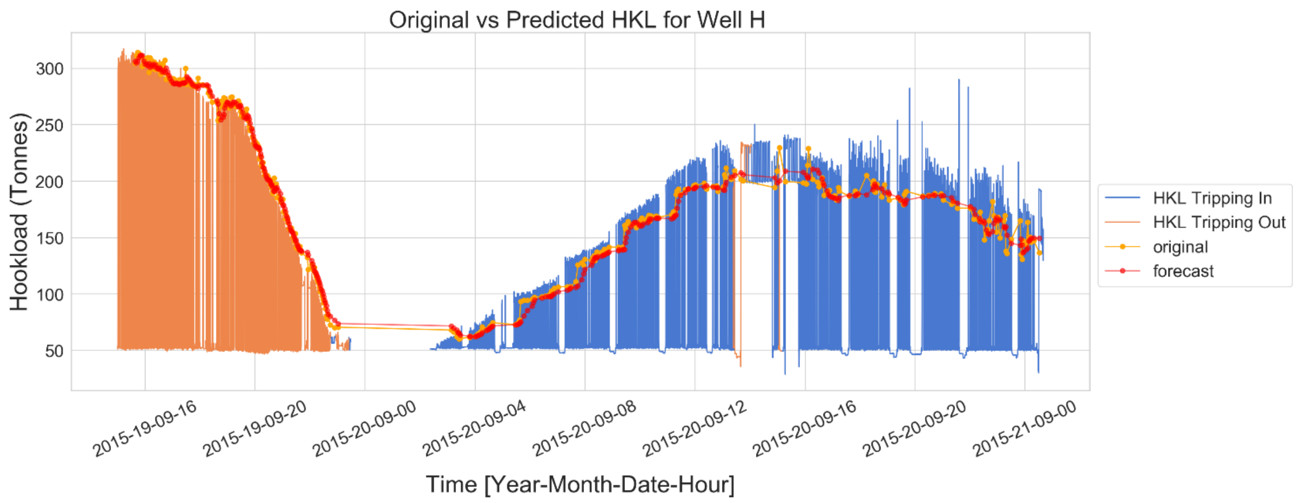

Figure 7.1: *Predicted vs Actual HKL values for Well H, where the whole HKL signal is included*

A closer examination of Figure 7.2 shows some obvious areas where the model predicts poorly. These are the zones between 2015-20-09-20 and 2015-20-09-00, as well as the area located between 2015-20-09-12 and 2015-20-09-16. The errors represented in these areas are mostly likely caused by deficiencies in the algorithm mentioned in chapter 6.3.1, which were developed to differentiate between the three operations. This can be confirmed by comparing Figure 7.1 to the second date-interval. For this area, there are some lines between the lowering operations which are categorized as tripping out. In reality, these lines should have been categorized as tripping in, which in return makes the models prediction more accurate than what the original features are proposing.

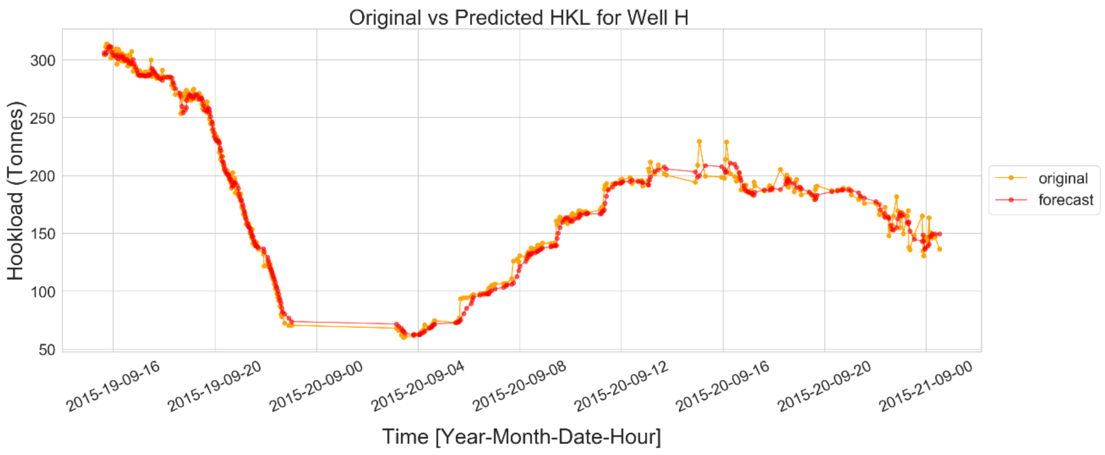

Figure 7.2: *Predicted vs Actual HKL values for Well H, where the whole HKL signal is excluded*

Besides areas where the model predicts poorly due faults in the developed model, one would also be interested in how well the overall predictions are. Figure 6.12 presents a function that was developed to extract random samples from the test data. There are 8 snippets presented, where each sample represent how well the model predicts against the original data for every 4th point in time. From these particular samples, the largest detected error between the original and forecasted data is approximately 10 tonnes. For Hookload data, an offset of 10 tonnes is considered rather small and feasible for Well H, as the WOH is observed to reach almost 400 tonnes.

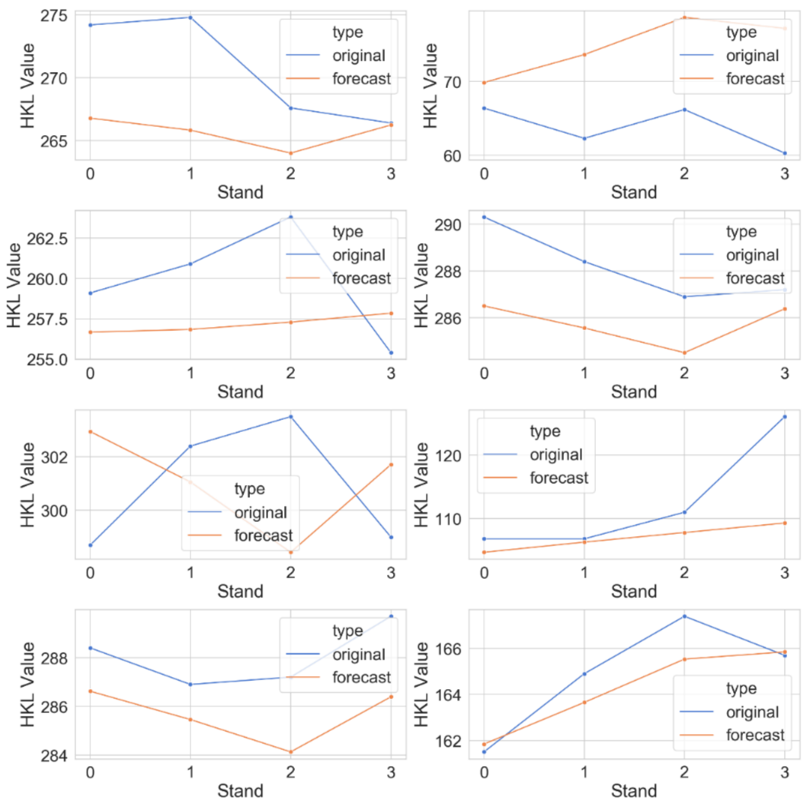

Figure 7.3: *Close samples of the model predictions vs the original data*

To determine how the error terms are distributed, the residuals of the model were also calculated as the difference between y_i and $\hat{y}_I$. A residual plot is a convenient way of determining the model's predictive capabilities on unseen data. If the residual plot contains a surplus of positive values, the model is generally overestimating its prediction, and if the plot contains a surplus of negative values, the model is generally underestimating its predictions. The residual plot for the developed ML model can be seen in Figure 7.4, where the value of errors are centered around zero, and mostly normally distributed down to 20 tonnes. This indicates that the model is capable of handling most changes in the unseen portion of the dataset, and that the predictions made does not contain any systematic offset.

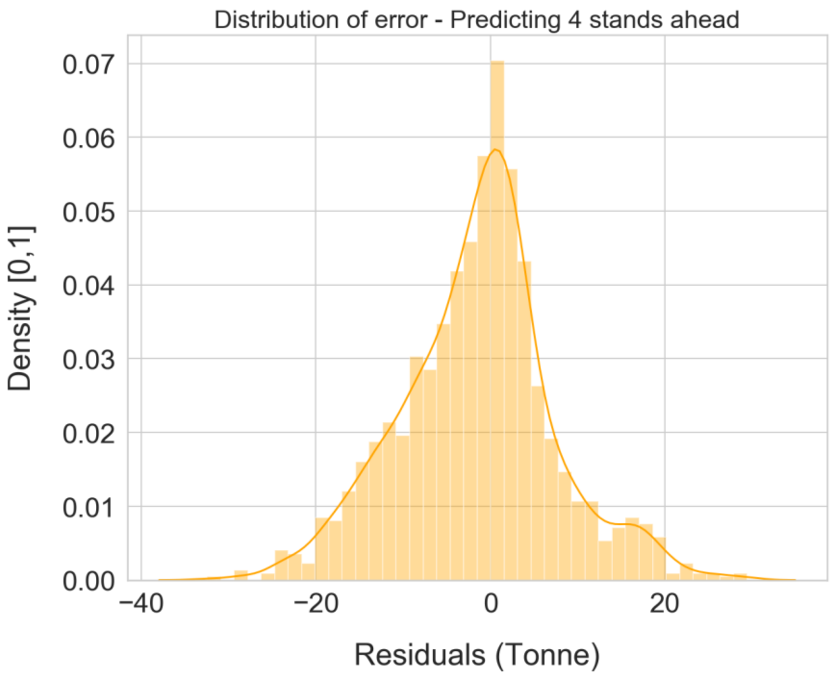

Figure 7.4: *Residual distributing for the LSTM model*

However, the left side of the of the plot do have a longer tail reaching -40 at the bottom. This visible tail is most likely also caused by the algorithm developed in chapter 6.3.1, as a longer negative tail implies that the predicted output is lower than the actual output, which is true for the interval between 2015-20-09-12 and 2015-20-09-16. Nevertheless. It is possible that the model is slightly underestimating its results, but based on the information provided by Figure 7.2 and Figure 7.3 this is highly unlikely. Lastly, Figure 7.5 shows the absolute distribution of errors of the test data from Well H. Out of the 367 points used for testing, almost 160 of the points were predicted with 0 errors. In addition, for the 367 datapoints in the test set, a MAE was 6.73 tonnes.

Despite having issues with wrongly categorized operations, as well as missing feature points for many of the stands, the results provided by the LSTM model is promising. Changes in wellbore trajectory and increased frictions also seems to have little impact on the model, as it was able to adapt to the changes in slope. In addition, using a single model for drilling, tripping in, and tripping out, as opposed to what is currently practiced by Exebenus, does also appear to be a viable option for stationary datasets.

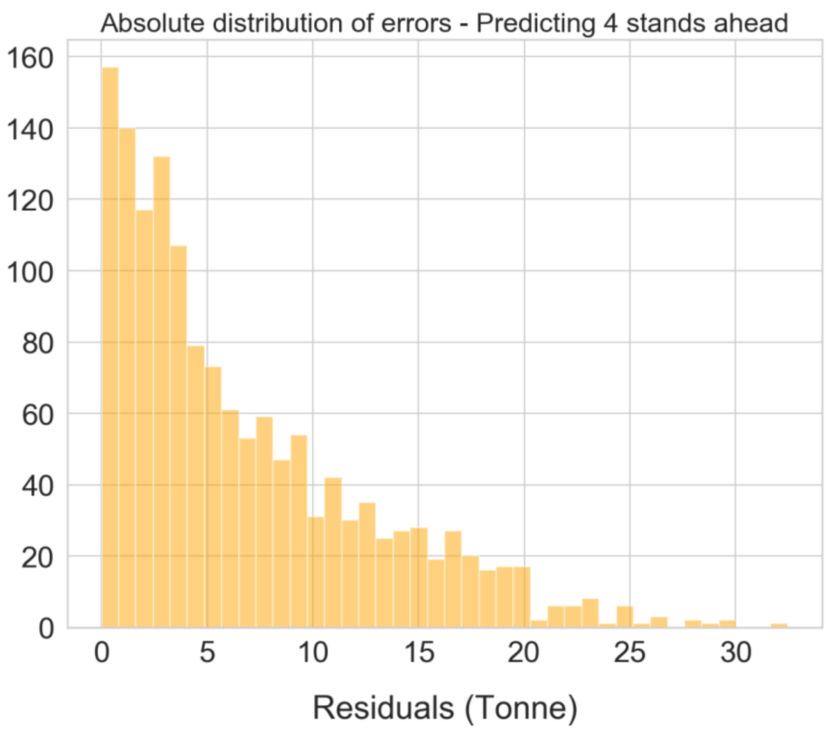

Figure 7.5: *Absolute distribution of errors in the ML model*

7.2 Importance of data quality:

Through the conducted research on machine learning and conversations with experts in this field, the overall quality of the data provided for ML project seems to be dependent on many factors. First of all, a ML model is reliant on large amounts of data to be trained properly. If not, there is a risk associated with applying the model to real-time operations, as a model trained on only a few datasets are more prone to predict inaccurately on unforeseen events. In addition, the reality seems to be that major portions of time spent on a ML project is used for data cleaning and filtering, rather than model development and performance evaluation. This is also the case throughout this thesis work, as a total of 18 wells were investigated to see if they were applicable to train the model. However, only two of them were sufficient for training purposes, which comprises the attained results to a certain extent.

Secondly, even if large amounts of data are available for disposal, the quality of the data utilized will also play a major role in the model performance. Sensor failures and drifts seems to be a reoccurring problem for many of the datasets. For personnel working with real-time drilling operations both -on an offshore, being able to continuously monitor rig-data is of importance in terms of preventing stuck pipe, as well as other situations. Because of this, a question is raised whether the sensors fail as often as the datasets imply, or if the files which are received for analytical purposes are corrupted somewhere along the way. Another possibility that might explain the overall trend of sensory failure could be that logging equipment is switched off when it is not needed, or that other type of equipment on the drill-site causes disturbances in the signal.

Furthermore, operational practices, as well as company procedures, also affect the quality of the data. In later conversations with Exebenus, it came to light that most of their clients work with one second frequency data for their vertical and deviated wells, but almost none their customers operate with the same frequency for the horizontal wells. The explanation behind this could in fact be that storage of high frequency data is still a challenge, and that the data is pre-filtered to compensate for lack of storage capacity. For the DS model, which uses the specific minimum or maximum HKL value after the slips are released, lack of dense data could make it difficult in terms of model performance, as the number of stands needed to readjust the model between each prediction would potentially increase, which could translate to errors in the stuck pipe classification model. However, if their model is implemented to real time data, it is believed that this issue will purely depend on how well the model is able to be trained on the 10 second frequency data.

8 Conclusion

In this research project provided by Exebenus, their machine learning regression model used to predict differential sticking in vertical wells were replicated, trained, and tested for a horizontal well to investigate the model's applicability. From the attained results, the following conclusions has been made:

- Frictional forces did impact the hookload in accordance to the researched theory. This behavior could be observed as a difference in the steepness of the slope between hoisting and lowering operations, where POOH increased the slope of the HKL value, while RIH operations reduced the slope of the HKL value. Regardless, the ML model was capable of adapting to the changes presented in the dataset.
- The model displays a residual distribution curve centered around 0, indicating that the model does not have any systematic offsets. Out of the 367 points used for testing, the model was capable of predicting roughly 160 points with 0 errors with a MAE of 6,73 tonnes. The predicted values which had larger deviations from the original values were mostly caused by faulty input data rather than poor model performance.
- Model training and testing were conducted using a single model to differentiate between drilling, tripping in and tripping out operations. This is unlike how Exebenus operates, as they are currently using two ML models to predict DS, one specified for drilling and one for tripping. The results have proven that a single model could be sufficient for all three operations, at least when working with existing time-series data.
- Due to the above-mentioned statements, it has been concluded that the regression model for DS developed by Exebenus in vertical sections, are also applicable to their longer, horizontal well section.

8.1 Future Work

The recommended future work surrounding the ML models provided by Exebenus is as follows:

- In terms of the DS model, one should investigate the disappearance of the distinct hookload signature for certain stands using higher frequency data, in order to clarify whether this issue is caused by low sampling frequency, or if there are other factors influencing the hookload during connection in lateral wells.
- Investigate if higher frequency data will better or worsen the accuracy of the DS model in horizontal well sections
- Further investigating if a single DS model is feasible for prediction of both drilling and tripping operations, as opposed to the two separate models which are currently used.
- Investigate how the input features used for the WG model and HC model are affected by friction, inclination and other parameters. In addition, investigation should also be onducted on how low frequency data can impact the predictive capabilities of these models.

9 References

ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G. S., DAVIS, A., DEAN, J. & DEVIN, M. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

ABBAS, A., FLORI, R., ALMUBARAK, H., DAWOOD, J., ABBAS, H. & ALSAEDI, A. 2019. Intelligent Prediction of Stuck Pipe Remediation Using Machine Learning Algorithms.

ALHASHEM, M. Supervised Machine Learning in Predicting Multiphase Flow Regimes in Horizontal Pipes, 10.2118/197545-ms. Abu Dhabi International Petroleum Exhibition & Conference, 2019. D021S043R003.

ALSHAIKH, A. A., ALBASSAM, M. K., AL GHARBI, S. H. & AL-YAMI, A. S. 2018. Detection of Stuck Pipe Early Signs and the Way Toward Automation. *Abu Dhabi International Petroleum Exhibition & Conference.* Abu Dhabi, UAE: Society of Petroleum Engineers.

ANSARI, A., FATHI, E., BELYADI, F., TAKBIRI-BORUJENI, A. & BELYADI, H. Data-Based Smart Model for Real Time Liquid Loading Diagnostics in Marcellus Shale via Machine Learning, 10.2118/189808-ms. SPE Canada Unconventional Resources Conference, 2018. D021S007R001.

ASSADI, M., SHAIPOV, I., AL FURATI, B. & GREGERSEN, J. M. 2019. Compendium in the course PET100 at the University of Stavanger. *Introduction to Drilling in the Petroleum Industry***,** 198.

AZOFF, E. M. 1993. Reducing error in neural network time series forecasting. *Neural Computing & Applications,* 1**,** 240-247.

BANGERT, P. 2021. Machine learning and data science in the oil and gas industry : best practices, tools, and case studies. Amsterdam: Gulf Professional Publishing.

BELLARBY, J. 2009. Well completion design. 1st ed. ed. Amsterdam, Netherlands: Elsevier.

BELYADI, H. & HAGHIGHAT, A. 2021. Machine learning guide for oil and gas using Python : a step-by-step breakdown with data, algorithms, codes, and applications. Amsterdam: Gulf Professional Publishing.

BENTEC 2017. Technical Bulletin on Top Drive TD-500-HT & TD-350-HT. Bentec GmbH Drilling & Oilfield Systems.

BOURGOYNE, A. T., MILLHEIM, K. K., CHENEVERT, M. E. & YOUNG, F. S. 1986. *Applied drilling engineering*, Society of Petroleum Engineers Richardson, TX.

BOWES, C. & PROCTER, R. 1997. *Drillers Stuck Pipe Handbook*, Procter & Collins Ltd.

BRADLEY, W. B., JARMAN, D., PLOTT, R. S., WOOD, R. D., SCHOFIELD, T. R., AUFLICK, R. A. & COCKING, D. 1991. A Task Force Approach to Reducing Stuck Pipe Costs. *SPE/IADC Drilling Conference.* Amsterdam, Netherlands: Society of Petroleum Engineers.

BRITANNICA, T. E. O. E. 2021. Friction. *In:* BRITANNICA, E. (ed.) *Britannica.*

BROWNIEE, J. 2019. *How to Choose Loss Functions When Training Deep Learning Neural Networks* [Online]. Available: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/ [Accessed 05.06 2021].

BROWNIEE, J. 2021. *How to Choose an Activation Function for Deep Learning* [Online]. Available: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/ [Accessed 10.04 2021].

BYROM, T. G. 2007. *Casing and Liners for Drilling and Completion,* Austin, Austin: Gulf Publishing Company.

CAYEUX, E., MESAGAN, T., TANRIPADA, S., ZIDAN, M. & FJELDE, K. K. K. Real-Time Evaluation of Hole-Cleaning Conditions With a Transient Cuttings-Transport Model, 10.2118/163492-pa. SPE Drilling & Completion, 2014. SPE-163492-PA, 5-21.

CAYEUX, E., SKADSEM, H. J. & KLUGE, R. Accuracy and Correction of Hook Load Measurements During Drilling Operations, 10.2118/173035-ms. SPE/IADC Drilling Conference and Exhibition, 2015. D021S010R005.

CHOLLET, F. 2015. *Keras* [Online]. GitHub. Available: https://github.com/fchollet/keras [Accessed 20.04 2021].

CRUMPTON, H. 2018. *Well Control for Completions and Interventions,* Saint Louis, Saint Louis: Elsevier Science & Technology.

DAGRAIN, F. 2001. Influence of the cutter geometry in rock cutting : an experiment approch.

DEEPAI. 2020. *Evalutation Metrics* [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/evaluation-metrics#:~:text=Evaluation%20metrics%20are%20used%20to,statistical%20or%20machine%20learning%20model.&text=There%20are%20many%20different%20types,%2C%20confusion%20matrix%2C%20and%20others. [Accessed].

DOSHI, M. 2014. Stuck up, Packing and Bridging.

FJELDE, K. K. 2019. Presentation in PET505 Course, Drilling & Operation. University of Stavanger.

FREISLEBEN, B. Stock market prediction with backpropagation networks. International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, 1992. Springer, 451-460.

GENÇAY, R. & QI, M. 2001. Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. *IEEE Transactions on Neural Networks,* 12**,** 726-734.

GLOMSTAD, T. S. 2012. *Analysis of Hook Load Signal to Reveal the Causes of Restrictions.* Master, Norwegian University of Science and Technology.

GOOGLE. 2021. *Machine Learning Crash Course* [Online]. Available: https://developers.google.com/machine-learning/crash-course [Accessed 10.02 2021].

GULSRUD, T. O., NYBØ, R. & BJØRKEVOLL, K. S. Statistical Method for Detection of Poor Hole Cleaning and Stuck Pipe, 10.2118/123374-ms. SPE Offshore Europe Oil and Gas Conference and Exhibition, 2009. SPE-123374-MS.

HAN, D., JUNG, J. & KWON, S. 2020. Comparative Study on Supervised Learning Models for Productivity Forecasting of Shale Reservoirs Based on a Data-Driven Approach. *Applied Sciences,* 10**,** 1267.

HEATON, J. 2008. Introduction to neural networks with Java, Heaton Research, Inc.

HOCHREITER, S. & SCHMIDHUBER, J. 1997. Long Short-Term Memory. *Neural Computation,* 9, 10.1162/neco.1997.9.8.1735**,** 1735-1780.

JOHANCSIK, C., FRIESEN, D. & DAWSON, R. 1984. Torque and drag in directional wells-prediction and measurement. *Journal of Petroleum Technology,* 36**,** 987-992.

KıZRAK, A. 2019. *Comparison of Activation Functions for Deep Neural Networks* [Online]. Available: https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a [Accessed 11.06 2021].

LUKE, G. R. & JUVKAM-WOLD, H. C. 1993. The Determination of True Hook-and-Line Tension Under Dynamic Conditions. *SPE Drilling & Completion,* 8, 10.2118/23859-PA**,** 259-264.

MAIDLA, E. & WOJTANOWICZ, A. Field comparison of 2-D and 3-D methods for the borehole friction evaluation in directional wells. SPE Annual Technical Conference and Exhibition, 1987. Society of Petroleum Engineers.

MCKINNEY, W. Data Structures for Statistical Computing in Python. *In:* VAN DER WALT, S. & MILLMAN, J., eds. Proceeding of the 9th Python Science Conference, 2010.

MEOR HASHIM, M. M., YUSOFF, M. H., ARRIFFIN, M. F., MOHAMAD, A., GOMES, D., JOSE, M. & TENGKU BIDIN, T. E. Utilizing Artificial Neural Network for Real-Time Prediction of Differential Sticking Symptoms, 10.2523/iptc-21221-ms. International Petroleum Technology Conference, 2021a. D071S027R002.

MEOR HASHIM, M. M., YUSOFF, M. H., ARRIFFIN, M. F., MOHAMAD, A., TENGKU BIDIN, T. E. & GOMES, D. Case Studies for the Successful Deployment of Wells Augmented Stuck Pipe Indicator in Wells Real Time Centre, 10.2523/iptc-21199-ms. International Petroleum Technology Conference, 2021b. D031S013R002.

MITCHELL, J. 2011. *Trouble-free Drilling*, Drilbert Engineering.

MITCHELL, R. F. & MISKA, S. 2011. *Fundamentals of drilling engineering*, Richardson, Tex, Society of Petroleum Engineers.

NKETAH, G. U. 2016. *Comparison of Cloud Machine Learning Services.* Master, University of Stavanger.

PAPAVINASAM, S. 2014. Corrosion control in the oil and gas industry. Boston: Elsevier.

PATHAK, M., COSBY, T. & PERRONS, R. K. 2021. Guest Editorial: Artificial Intelligence in the E&amp;P Industry: What Have We Learned So Far and Where to Next? *Journal of Petroleum Technology,* 73, 10.2118/0121-0012-jpt**,** 12-13.

PETROPEDIA. 2021. *Unconsolidated Formation* [Online]. Petropedia. Available: https://www.petropedia.com/definition/9675/unconsolidated-formation/ [Accessed 25.02 2021].

POLAK, M. A. & LASHEEN, A. 2001. Mechanical modelling for pipes in horizontal directional drilling. *Tunnelling and Underground Space Technology,* 16, https://doi.org/10.1016/S0886-7798(02)00020-2**,** 47-55.

RASCHKA, S. & MIRJALILI, V. 2019. *Python Machine Learning, 3rd Ed.*, Packt Publishing.

SACHDEV, H. S. 2020. *Choosing Number of Hidden Layers and Number of Hidden Neurons in Neural Networks* [Online]. LinkeDn. Available: https://www.linkedin.com/pulse/choosing-number-hidden-layers-neurons-neural-networks-sachdev/ [Accessed 05.03 2021].

SALMINEN, K., CHEATHAM, C., SMITH, M. & VALIULLIN, K. 2017. Stuck-Pipe Prediction by Use of Automated Real-Time Modeling and Data Analysis. *SPE Drilling & Completion,* 32, 10.2118/178888-pa**,** 184-193.

SCHLUMBERGER. 2020. *Hook Load* [Online]. Schlumberger Oil Field Glossary: Schlumberger. Available: https://www.glossary.oilfield.slb.com/en/terms/h/hook_load [Accessed 27.11 2020].

SCHLUMBERGER. 2021a. *Casing String* [Online]. Schlumberger Oilfield Glossary. Available: https://www.glossary.oilfield.slb.com/en/terms/c/casing_string [Accessed 06,03 2021].

SCHLUMBERGER. 2021b. *Impermeable* [Online]. Schlumberger Oilfield Glossary: Schlumberger. Available: https://www.glossary.oilfield.slb.com/en/terms/i/impermeable [Accessed 26.02 2021].

SCHLUMBERGER. 2021c. *Jet* [Online]. Schlumberger Oilfield Glossary: Schlumberger. Available: https://www.glossary.oilfield.slb.com/en/Terms/j/jet.aspx [Accessed 05.03 2021].

SCHLUMBERGER. 2021d. *Make a Connection* [Online]. Schlumberger Oilfield Glossary. Available: https://www.glossary.oilfield.slb.com/en/Terms/m/make_a_connection.aspx#:~:text=To%20add%20a%20length%20of,as%20the%20well%20is%20drilled. [Accessed 05.03 2021].

SCHLUMBERGER. 2021e. *MWD* [Online]. Schlumberger Oilfield Glossary: Schlumberger. Available: https://www.glossary.oilfield.slb.com/en/terms/m/mwd [Accessed 22.03 2021].

SCHLUMBERGER. 2021f. *Overpressure* [Online]. Schlumberger Oil Field Glossary: Schlumberger. Available: https://www.glossary.oilfield.slb.com/en/Terms/o/overpressure.aspx#:~:text=Subsurface%20pressure%20that%20is%20abnormally,pressure%20at%20a%20given%20depth.&text=Abnormally%20high%20pore%20pressure%20can,fluids%20increases%20as%20overburden%20increases [Accessed 25.02 2021].

SCHLUMBERGER. 2021g. *Stand* [Online]. Schlumberger Oilfield Glossary. Available: https://www.glossary.oilfield.slb.com/en/terms/s/stand [Accessed 09.03 2021].

SHRIVASTAVA, A. 2018. 3 - Plastic Properties and Testing. *In:* SHRIVASTAVA, A. (ed.) *Introduction to Plastics Engineering.* William Andrew Publishing.

SINGH, H., SEOL, Y. & MYSHAKIN, E. M. 2020. Automated Well-Log Processing and Lithology Classification by Identifying Optimal Features Through Unsupervised and Supervised Machine-Learning Algorithms. *SPE Journal*, 25, 10.2118/202477-pa**,** 2778-2800.

STRAND, E. 2014. Hydraulic Calculations Using Discovery Web for Visualization. Master, University of Stavanger.

TORBERGSEN, H.-E. B., HAGA, H. B., SANGESLAND, S., AADNØY, B. S., SÆBY, J., JOHNSEN, S., RAUSAND, M. & LUNDETEIGEN, M. A. 2012. Introduction to Well Integrity. *In:* NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY, NORSK OLJE & GASS & UNIVERSITY OF STAVANGER (eds.).

TVEITDAL, T. 2011. Toruque & Drag Analyses of North Sea Wells Using New 3D Model. Master's University of Stavanger.

WARREN, J. D. 2010. *The Benefits of Top Drive Drilling* [Online]. National Oilwell Varco. Available: http://dthrotarydrilling.com/News/3-Feb-2010/Top_Drive_Drilling.html [Accessed 23.03 2021].

WILLIAM, P. 17.12 2020. Powerlaw Fluids. *Beyond Discovery* [Online]. Available from: https://www.beyonddiscovery.org/bingham-plastic/321-powerlaw-fluids.html [Accessed 06.05 2021].

ZHANG, A., LIPTON, Z. C., LI, M. & SMOLA, A. J. 2020. Dive into Deep Learning.

ZHENG, H. & WU, Y. 2019. A XGBoost Model with Weather Similarity Analysis and Feature Engineering for Short-Term Wind Power Forecasting. *Applied Sciences,* 9, 10.3390/app9153019.

AADNOY, B. S. 2011. Modern well design. 2nd ed. ed. London,New York: CRC Press/Balkema.

AADNOY, B. S. & KAARSTAD, E. Theory and Application of Buoyancy in Wells, 10.2118/101795-ms. IADC/SPE Asia Pacific Drilling Technology Conference and Exhibition, 2006. SPE-101795-MS.

APPENDIX A: Johancsik T&D Model

Simplified Johancsik T&D Model

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
sns.set(style = 'whitegrid')

print('Setup Complete')
```

```
Setup Complete
```

image.png

image.png

```
def Johancisk_Model(
    rho_mud = 1.7,
    rho_steel = 7.85,
    W_drillstring = 31.06,
    F1 = 0,
    muy = 0.3,
    DLS = 3/30,
    L_Stand = 30,
    theta=0,
    theta_build_1 = 0,
    theta_build_2 = 90,
    build_section = 900,
    Inclined_section = 1800,
    horizontal_section = 900):

    """
    rho_mud: mud density [sg]
    rho_steel: pipe density [sg]
    W_drillstring: Drillstring Weight [Kg/m]
    F1: Force at bottom of string [N]
    muy: Friction Coefficient
    DLS: Dogleg Severity [deg/m]
    L_Stand: Length of drillpipe stand [m]
    theta: Angle at inclined/Vertical section [deg]
    theta_build_1: Angle at start of Build up [deg]
    theta_build_2: Angle at bottom of Build up [deg]
    build_section: Length of Build Section [m]
    kickoff_section: Length of vertical/Inclined Section [m]
    horizontal_section: Length of Horizontal section [m]
    """

    #Calc Boyancy
    Buoyancy= 1-(rho_mud/rho_steel)
    w_ds = (Buoyancy*W_drillstring*9.81)/1000
    F1 = F1/1000

    kickoff = Inclined_section
    end_ARC = Inclined_section + build_section
    horizontal = Inclined_section + build_section + horizontal_section
    dtheta = DLS*L_Stand
    theta = (theta * np.pi)/180

    lst_tripping_in  = []
    lst_tripping_out = []
    lst_static       = []
    lst_depth        = []

    depth = 0
    #Vertical Section
    while depth < kickoff:

        tripping_in = F1 + ( (w_ds * depth) * ( np.cos(theta) - muy*np.sin
(theta) ) )
        tripping_out = F1 + ( (w_ds * depth) * ( np.cos(theta) + muy*np.si
n(theta) ) )
        static = F1 + (w_ds*depth*np.cos(theta) )

        lst_tripping_in.append(tripping_in)
        lst_tripping_out.append(tripping_out)
        lst_static.append(static)
        lst_depth.append(depth)

        depth += L_Stand

    #Build Section
    theta2  = theta_build_1 * (np.pi/180)
    theta1  = theta_build_1 + (DLS*L_Stand) * (np.pi/180)
    while depth >= kickoff and depth < end_ARC and theta_build_2 > theta1:

        #Converting to Radians
        t2 = theta2 * (np.pi/180)
        t1 = theta1 * (np.pi/180)

        #Using Avg angle method
        average = ( t2 + t1 )/2

        delta_theta = np.sin(t1) - np.sin(t2)

        tripping_in_build = F1 + ( w_ds*depth*np.cos(average) ) - muy*( ((
F1*delta_theta)
                                                                   + (w_ds*de
pth*np.sin(average)))**2 )**(1/2)

        tripping_out_build = F1 + ( w_ds*depth*np.cos(average) ) + muy*( (
(F1*delta_theta)
                                                                    + (w_ds*d
epth*np.sin(average)))**2 )**(1/2)
```

```
        static_build = F1 + ( w_ds*depth*np.cos(average) )

        lst_tripping_in.append(tripping_in_build)
        lst_tripping_out.append(tripping_out_build)
        lst_static.append(static_build)
        lst_depth.append(depth)

        depth  += L_Stand
        theta2 += DLS*L_Stand


    #Horizontal Section
    Horizontal_Distance = -0
    while depth >= end_ARC and depth < horizontal:

        theta_2 = theta_build_2* (np.pi/180)

        tripping_in_hor = F1 - (w_ds * Horizontal_Distance *muy*np.sin(the
ta_2)) + tripping_in_build
        tripping_out_hor = F1 + (w_ds * Horizontal_Distance * muy*np.sin(t
heta_2)) + tripping_out_build
        static_hor = F1 + static_build

        lst_tripping_in.append(tripping_in_hor)
        lst_tripping_out.append(tripping_out_hor)
        lst_static.append(static_hor)
        lst_depth.append(depth)

        depth += L_Stand
        Horizontal_Distance += L_Stand

    plt.figure(figsize=(6,8))
    plt.plot(lst_tripping_out,lst_depth,label="Hoisting",color="Red")
    plt.plot(lst_tripping_in,lst_depth,label="Lowering",color="g")
    plt.plot(lst_static, lst_depth, label = 'Static')
    plt.title("Johancsik Torque & Drag Model", fontsize=15)
    plt.xlabel("\nWOH (kN)", fontsize=15)
    plt.ylabel("\nDepth (mMD)", fontsize=15)
    plt.gca().invert_yaxis()
    plt.legend()
    plt.grid(True)
    plt.show()

Johancisk_Model(
    rho_mud = 1.7,
    rho_steel = 7.85,
    W_drillstring = 31.06,
    F1 = 0,
    muy = 0.3,
    DLS = 3/30,
    L_Stand = 30,
    theta=0,
    theta_build_1 = 0,
    theta_build_2 = 90,
    build_section = 900,
    Inclined_section = 900,
    horizontal_section = 900)
```

APPENDIX B: ML MODEL

```
#!pip install tensorflow==2.1.0
import numpy as np
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import seaborn as sns
from pylab import rcParams
import matplotlib.pyplot as plt
from matplotlib import rc
from sklearn.model_selection import train_test_split
from pandas.plotting import register_matplotlib_converters
import matplotlib.pyplot as plt
from scipy.signal import find_peaks
from scipy.signal import savgol_filter
import tensorflow as tf
from pandas.plotting import autocorrelation_plot
from matplotlib.dates import DateFormatter
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
%matplotlib inline

register_matplotlib_converters()
sns.set(style = 'whitegrid', palette ='muted', font_scale =2)
%config InlineBackend.figure_format = 'retina'

RANDOM_SEED = 42

np.random.seed(RANDOM_SEED)
tf.random.set_seed(RANDOM_SEED)
print(tf.__version__)
print('Setup Complete')
```

```
2.3.0
Setup Complete
```

```
#Read CSV File
read = pd.read_csv(r'C:\Users\Winu\Documents\Fag\Master-Thesis\EXEBENUSDAT
A\Well H.csv', error_bad_lines=False)

print(read.columns.tolist())

read['Drilling Activity'].value_counts()
```

```
0    161886
Name: Drilling Activity, dtype: int64
```

```
#Only Include relevant Columns:
df = pd.read_csv(r'C:\Users\Winu\Documents\Fag\Master-Thesis\EXEBENUSDATA\
Well H.csv',
                 error_bad_lines=False, usecols = ['Hole Depth', 'Bit Dept
h', 'True Vertical Depth', 'Hook Load', 'Block Height','Standpipe Pressure
', 'Inclination', 'Min Hook Load','TORQUE (ft/lbs)', 'Drilling Activity',
'Flow','YYYY/MM/DD', 'HH:MM:SS', 'Memos'],parse_dates = [['YYYY/MM/DD', 'H
H:MM:SS']])

#Concatenating the time column for plotting purposes
df.rename(columns={'YYYY/MM/DD_HH:MM:SS': 'Time'}, inplace=True)

#Setting New Index
df.set_index('Time', inplace = True)
```

Feature Engineering:

```
#Check columns
print(df.columns.tolist())
```

```
['Hole Depth', 'Bit Depth', 'True Vertical Depth', 'Hook Load', 'Standpipe
 Pressure', 'Flow', 'Block Height', 'Inclination', 'Min Hook Load', 'TORQU
E (ft/lbs)', 'Drilling Activity', 'Memos']
```

```
df.head()
```

```
df.rename(columns = {'TORQUE (ft/lbs)':'tqa'}, inplace = True)
df['seconds'] = df.index.second
df['minutes'] = df.index.minute
df['hour'] = df.index.hour
df['day_of_month'] = df.index.day
df['month'] = df.index.month

hourly = df.resample('T').sum()
```

```
#Plotting Raw HKL Values:
fig, ax = plt.subplots(figsize=(20, 8))
fig.tight_layout(pad=4.0)
date_form = DateFormatter("%Y-%d-%m")
ax.plot(df['Hook Load'], label='Hookload')
ax.set_title('Hookload Data for Well H', fontsize=30)
ax.set_ylabel('Hookload (Tonnes)', fontsize=30)
ax.xaxis.set_major_formatter(date_form)
ax.tick_params(axis='x', labelrotation=25)
ax1 = ax.twinx()
ax1.plot(df['Inclination'], '--', color='red', label = 'Inclination')

ax.legend(bbox_to_anchor = [0.2,1.15])
ax1.legend(bbox_to_anchor = [1,1.15])

plt.show()
```

```
print('Start Date:', df.index.min())
print('\nEnd Date:', df.index.max())

df.count()
```

```
Start Date: 2015-09-03 03:41:40

End Date: 2015-09-21 21:51:50

Hole Depth             162060
```

```
Bit Depth                 162060
True Vertical Depth       162060
Hook Load                 162060
Standpipe Pressure        162060
Flow                      162060
Block Height              162060
Inclination               162060
Min Hook Load             162060
tqa                       162060
Drilling Activity         162060
Memos                       1470
seconds                   162060
minutes                   162060
hour                      162060
day_of_month              162060
month                     162060
dtype: int64

#Plotting Raw Depth Measurements
fig, ax = plt.subplots(figsize=(20, 8))
fig.tight_layout(pad=4.0)
date_form = DateFormatter("%Y-%d-%m")
ax.plot(df['Hole Depth'], color='Red', label='Hole Depth (mD)')
ax.plot(df['Bit Depth'], '-.', color='Black', label='Bit Depth (mD)')
ax.xaxis.set_major_formatter(date_form)
ax.tick_params(axis='x', labelrotation=25)
ax.invert_yaxis()
ax.set_title('Depth Measurements for Well H', fontsize=30)
ax.set_ylabel('Depth [ft]', fontsize=30)
ax.legend(loc = 'best') #loc=bbox"upper left")

plt.show()

df['Bit Depth'].head(60)

#Removing outliers at beginning of sampling interval:
df = df.drop(df.index[0:53])

#Check if outliers are removed:
df.head()

#Plotting Bit Depth and Hole Depth again:

fig, ax = plt.subplots(figsize=(20, 8))
fig.tight_layout(pad=4.0)
date_form = DateFormatter("%Y-%d-%m")
ax.plot(df['Hole Depth'], color='Red', label='Hole Depth (mD)')
ax.plot(df['Bit Depth'], '-.', color='Black', label='Bit Depth (mD)')
ax.xaxis.set_major_formatter(date_form)
ax.tick_params(axis='x', labelrotation=25)
ax.invert_yaxis()
ax.set_title('Depth Measurements for Well H', fontsize=30)
ax.set_ylabel('Depth [ft]', fontsize=30)
ax.legend(loc = 'best') #loc=bbox"upper left")

plt.show()

print(df.loc[df['Hook Load'] < 100].mean())

Hole Depth                9032.623616
Bit Depth                 5892.113397
True Vertical Depth       1268.663060
Hook Load                   52.311738
Standpipe Pressure          86.685774
Flow                      -235.462078
Block Height                42.009521
Inclination                 35.357828
Min Hook Load               50.504415
tqa                        645.421090
Drilling Activity            0.000000
seconds                     25.021140
minutes                     29.521215
hour                        11.390706
day_of_month                11.535153
month                        9.000000
dtype: float64

#Removing HKL Outliers
df['2015-09-08']['Hook Load'].values[df['2015-09-08']['Hook Load'] < 50] =
 50

#Removing Inclination Outlier:
df['Inclination'].values[(df['Inclination'].values > 90) | (df['Inclinatio
n'].values == 0 )] = 90

#Plotting To check if issues are fixed:
fig, ax = plt.subplots(figsize=(20, 8))
fig.tight_layout(pad=4.0)
date_form = DateFormatter("%Y-%d-%m")
ax.plot(df['Hook Load'], label='Hookload')
ax.set_title('Hookload Data for Well H', fontsize=30)
ax.set_ylabel('Hookload (Tonnes)', fontsize=30)
ax.xaxis.set_major_formatter(date_form)
ax.tick_params(axis='x', labelrotation=25)
ax1 = ax.twinx()
ax1.plot(df['Inclination'], '--', color='red', label = 'Inclination')

ax.legend(bbox_to_anchor = [0.2,1.15]) #loc=bbox"upper left")
ax1.legend(bbox_to_anchor = [1,1.15]) #loc="upper right")

plt.show()

#Grouping Data in days to be able to investigate HKL values better
grouped = df.groupby(df.index.date)

ncols=1
nrows = int(np.ceil(grouped.ngroups))
```

```
fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(20,150), share
y=False)


for (key, ax) in zip(grouped.groups.keys(), axes.flatten()):
    grouped.get_group(key)['Hook Load'].plot(ax=ax, title=key)
    ax2 = ax.twinx()

plt.subplots_adjust(top = 0.99, bottom=0.01, hspace=0.5, wspace=0.1)
ax.legend()
plt.show()

#Differentiating Between Operations:

#New Column With Bit Depth Shifted 30 minutes Ahead;
df['Bit_Depth_next'] = df['Bit Depth'].shift(-240)

#New Column With Hole Depth Shifted 30 minutes Ahead:
df['Hole_Depth_next'] = df['Hole Depth'].shift(-240)

#Conditions For Drilling:
#Column where mD = Bit Depth
df['dnow'] = np.where( df['Hole Depth'] == df['Bit Depth'], True, False)

#Column where mD = Bit Depth 30 min ahead
df['dnext'] = np.where(df['Hole_Depth_next'] == df['Bit_Depth_next'], True
, False)

#Conditions for Tripping in
trip_in = (
( df['Hole Depth'] > df['Bit Depth'])
                                   & (df['Bit_Depth_next'] > df['Bit Depth
']))
#Conditions for Tripping Out:
trip_out = (
    ( df['Hole Depth'] > df['Bit Depth'])
                                    & (df['Bit_Depth_next'] < df['Bit Dept
h']))

# Finding all HKL values for Tripping OUT
df['Tripping_Out'] = df['Hook Load'].where(trip_out)

# Finding all HKLS values for Tripping IN
df['Tripping_In'] = df['Hook Load'].where(trip_in)

# Finding all HKL values for stands tripped in during Drilling:
df['Drilling'] = df['Hook Load'].where(((df['dnow'] < trip_in) &
                                                     (trip_in <= df['dnex
t']) == True))

#Remove all tripping in scenarios where Drilling is included:

df['Tripping_In'] = df['Tripping_In'].loc[(df['Tripping_In'] != (df['Drill
ing']))]
```

```
df['Tripping_Out'] = df['Tripping_Out'].loc[(df['Tripping_Out'] != df['Dri
lling'])]

# Plotting 30 min Interval
fig, ax = plt.subplots(figsize=(20, 8))
fig.tight_layout(pad=4.0)
date_form = DateFormatter("%Y-%d-%m")
ax.plot(df['Drilling'], label = 'Drilling')
ax.plot(df['Tripping_Out'], label = 'Tripping Out')
ax.plot(df['Tripping_In'], label = 'Tripping In')
ax.set_title('Separated Hookload Data for Well H, 30 minute interval', fon
tsize=30)
ax.set_ylabel('Hookload (Tonnes)', fontsize=30)
ax.xaxis.set_major_formatter(date_form)
ax.tick_params(axis='x', labelrotation=25)
ax.legend(loc='best')
```

```
<matplotlib.legend.Legend at 0x130c69384c8>
```

```
grouped = df.groupby(df.index.date)

ncols=1
nrows = int(np.ceil(grouped.ngroups))

fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(20,100), share
y=False)


for (key, ax) in zip(grouped.groups.keys(), axes.flatten()):

    grouped.get_group(key)['Drilling'].plot(ax=ax, title=key)
    grouped.get_group(key)['Tripping_Out'].plot(ax=ax, title=key)
    grouped.get_group(key)['Tripping_In'].plot(ax=ax, title=key)
    ax.set_title('HookLoad for Well H')
    ax.legend(loc='best')

plt.subplots_adjust(top = 0.99, bottom=0.01, hspace=0.5, wspace=0.1)
plt.show()

# Handling Double Indexing Issue, only saves last instance of the reoccuri
ng index value
def remove_double_index(seq):
    """Reverses a given list of values
    and keeps each first instance of a double index that occurs.
    Then, the list is tranformed back to its initial state"""

    seen = set()
    seen_add = seen.add
    return [x for x in seq[::-1] if not (x in seen or seen_add(x))][::-1]

#Finding Min and Max Values for the Different Scnearios:

#Drilling:
```

```
#Finding Min Values for HKL Drilling
df['HKLminDrill'] = df.iloc[find_peaks(-df['Drilling'],height = -60, thres
hold = None,
                                distance= 12)[0]]['Drilling']

#Finding all Minimum Values above a certain HKL Drilling value:
df['HKLmaxDrill'] = df.iloc[find_peaks(df['Drilling'], height=(80), thresh
old = None,
                                 distance= 6)[0]]['Drilling']

#Tripping in:
#Finding Min Values for HKL when tripping In:
df['HKLminTrip_in'] = df.iloc[find_peaks(-df['Tripping_In'],height = -60,
threshold = None,
                                distance= 9)[0]]['Tripping_In']

#Finding all Max Values for tripping In:
df['HKLmaxTrip_in'] = df.iloc[find_peaks(df['Tripping_In'], height=(55), t
hreshold = 0,
                                 distance= 1)[0]]['Tripping_In']

#Detecing all minimum values to find Local minima during tripping in:
df['HKLminTrip_in_all'] = df.iloc[find_peaks(-df['Tripping_In'],height = N
one, threshold = None,
                                  distance= 1)[0]]['Tripping_In']

#Tripping Out:
#Finding Min Values for HKL when tripping Out:
df['HKLminTrip_out'] = df.iloc[find_peaks(-df['Tripping_Out'],height = -60
, threshold = None,
                                distance= 9)[0]]['Tripping_Out']

#Finding all Max Values for tripping Out:
df['HKLmaxTrip_out'] = df.iloc[find_peaks(df['Tripping_Out'], height=(60),
 threshold = None,
                                 distance= 6)[0]]['Tripping_Out']


#Extracting our desired points:

#Drilling:
idx1 = []
idx2 = []
for i in df['HKLminDrill'][df['HKLminDrill'].notnull()].keys():
#     print("Min Drilling:",i)
    for j in df['HKLmaxDrill'][df['HKLmaxDrill'].notnull()].keys():
        if i < j:
#             print("Max Drilling:",j)
            idx1.append(j)
            break

#Tripping Out:
for i in df['HKLminTrip_out'][df['HKLminTrip_out'].notnull()].keys():
#     print("Min Tripping Out:",i)
```

```
    for j in df['HKLmaxTrip_out'][df['HKLmaxTrip_out'].notnull()].keys():
        if i < j:
#               print("Max Tripping Out:",j)
            idx1.append(j)
            break

#Tripping in:
for i in df['HKLminTrip_in'][df['HKLminTrip_in'].notnull()].keys():
#     print("Min Tripping Out:",i)
    for j in df['HKLmaxTrip_in'][df['HKLmaxTrip_in'].notnull()].keys():
        if i < j:
#               print('Max after min', j)
            for k in df['HKLminTrip_in_all'][df['HKLminTrip_in_all'].notnu
ll()].keys():
                if j < k:
#                       print("Local Min tripping in::",k)
                    idx2.append(k)
                    break
            break
#Removing Double Indexing for Tripping Out and Drilling
idx_filtered1 = remove_double_index(idx1)

#Removing Double indexing for Tripping In
idx_filtered2 = remove_double_index(idx2)

#Get HKL Values for Tripping Out and Drilling for the extracted indexes
df['HKL Features1'] = df['Hook Load'][idx_filtered1]

#Get HKL Values for Tripping In for the extracted indexes
df['HKL Features2'] = df['Hook Load'][idx_filtered2]

print('Number of Feature values:', df[['HKL Features1','HKL Features2']].c
ount())

Number of Feature values: HKL Features1     666
HKL Features2    1071
dtype: int64

df['HKL Features1'].dropna(inplace=True)

df['HKL Features2'].dropna(inplace=True)

print('Number of Feature values:', df[['HKL Features1','HKL Features2']].c
ount())

Number of Feature values: HKL Features1     666
HKL Features2    1071
dtype: int64

grouped = df.groupby(df.index.date)

ncols=1
nrows = int(np.ceil(grouped.ngroups))

fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(20,150), share
y=False, sharex=False)
```

```
for (key, ax) in zip(grouped.groups.keys(), axes.flatten()):
    grouped.get_group(key)['Drilling'].plot(ax=ax, title=key)
    grouped.get_group(key)['Tripping_Out'].plot(ax=ax, title=key)
    grouped.get_group(key)['Tripping_In'].plot(ax=ax, title=key)
    grouped.get_group(key)['HKL Features1'].plot(ax=ax, marker = 'o', colo
r='red', title=key)
    grouped.get_group(key)['HKL Features2'].plot(ax=ax, marker = 'o', colo
r='red', title=key)

plt.subplots_adjust(top = 0.99, bottom=0.01, hspace=0.5, wspace=0.1)
ax.legend()
plt.show()
```

png

```
fig, ax = plt.subplots(figsize=(20, 8))
fig.tight_layout(pad=4.0)
date_form = DateFormatter("%H:%M:%S")
ax.plot(df['2015-09-20']['Hook Load'][5000:6200], label='Hookload')
```

```
fig, ax = plt.subplots(figsize=(20, 8))
fig.tight_layout(pad=4.0)
date_form = DateFormatter("%H:%M:%S")
ax.plot(df['2015-09-20']['Hook Load'][8000:8150], label='Hookload')
ax.plot(df['2015-09-20']['HKL Features2'][8000:8150], marker = 'o', color=
'red', label='HKL Features')
ax.set_title('Snippet of Hookload Data for Well H', fontsize=30)
ax.set_ylabel('Hookload (Tonnes)', fontsize=30)
ax.set_xlabel('Time [Hour:Minute:Second]', fontsize=30, labelpad = 20)
ax.xaxis.set_major_formatter(date_form)
ax.legend(bbox_to_anchor=(1,0.6))
```

```
df['2015-09-20']['HKL Features2'].min(), df['2015-09-20']['HKL Features2']
.max()
```

```
(49.6, 229.1)
```

```
#Renmoving wrongly extracted points, i.e removing all min values that are
below 55klbm:

df['HKL Features2'] = df['HKL Features2'].where(df['HKL Features2'] > 60)

print('Number of available features after removal of faulty points:', df[[
'HKL Features1','HKL Features2']].count())
```

```
Number of available features after removal of faulty points: HKL Features1
    666
HKL Features2    556
dtype: int64
```

```
fig, ax = plt.subplots(figsize=(20, 8))
fig.tight_layout(pad=4.0)
date_form = DateFormatter("%H:%M:%S")
ax.plot(df['2015-09-20']['Hook Load'][8000:8150], label='Hookload')
ax.plot(df['2015-09-20']['HKL Features2'][8000:8150], marker = 'o', color=
'red', label='HKL Features')
ax.set_title('Snippet of Hookload Data for Well H', fontsize=30)
ax.set_ylabel('Hookload (Tonnes)', fontsize=30)
ax.set_xlabel('Time [Hour:Minute:Second]', fontsize=30, labelpad = 20)
ax.xaxis.set_major_formatter(date_form)
ax.legend(bbox_to_anchor=(1,0.6))

sum_features = df['Hook Load'].where( (df['Hook Load'] == df['HKL Features
1']) | (df['Hook Load'] == df['HKL Features2']))
df['HKL Features'] = sum_features

df['HKL Features'].count()

1222

# Check for Whole dataset:

grouped = df.groupby(df.index.date)

ncols=1
nrows = int(np.ceil(grouped.ngroups))

fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(20,150), share
y=False)


for (key, ax) in zip(grouped.groups.keys(), axes.flatten()):
    grouped.get_group(key)['Drilling'].plot(ax=ax, title=key)
    grouped.get_group(key)['Tripping_Out'].plot(ax=ax, title=key)
    grouped.get_group(key)['Tripping_In'].plot(ax=ax, title=key)


plt.subplots_adjust(top = 0.99, bottom=0.01, hspace=0.5, wspace=0.1)
ax.legend()
plt.show()

fig, ax = plt.subplots(figsize=(20, 8))
fig.tight_layout(pad=4.0)
date_form = DateFormatter("%H:%M:%S")
ax.plot(df['Hook Load']['2015-09-19'][6000:6500], '-', label = 'Hook Load'
)
ax.plot(df['HKL Features']['2015-09-19'][6000:6500], 'o', color='red', lab
el = 'HKL Features')
ax.set_title('Snippet of Hookload Data for Well H, Tripping Out', fontsize
=30)
ax.set_ylabel('Hookload (Tonnes)', fontsize=30)
ax.set_xlabel('Time [Hour:Minute:Second]', fontsize=30, labelpad = 20)
ax.xaxis.set_major_formatter(date_form)
ax.legend(bbox_to_anchor=(1,0.6))
```

ML Model:

```
# Get the HKL Features as a separate DF:
WellH = pd.DataFrame(df['HKL Features'], index = df.index)
```

```
#Remove NaN Values:
WellH.dropna(inplace=True)

#Check:
WellH.head()

print(WellH['HKL Features'].max(), WellH['HKL Features'].min())
```

```
368.3 60.0
```

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Sequential, load_model, Model
from tensorflow.keras.layers import Dense, LSTM, Dropout, LeakyReLU, Input
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.preprocessing import MinMaxScaler
import datetime
import math
```

Moving Window Function:

```
def create_X_Y(ts: np.array, lag=1, n_ahead=1, target_index=0) -> tuple:
    """
    A method to create X and Y matrices from a time series array for the t
raining of
    deep learning models
    """
    # Extracting the number of features that are passed from the array
    n_features = ts.shape[1]

    # Creating placeholder lists
    X, Y = [], []

    if len(ts) - lag <= 0:
        X.append(ts)
    else:
        for i in range(len(ts) - lag - n_ahead):
            Y.append(ts[(i + lag):(i + lag + n_ahead), target_index])
            X.append(ts[i:(i + lag)])

    X, Y = np.array(X), np.array(Y)

    # Reshaping the X array to an RNN input shape
    X = np.reshape(X, (X.shape[0], lag, n_features))

    return X, Y
```

LSTM Model:

```
class NNMultistepModel():

    def __init__(
        self,
        X,
        Y,
        n_outputs,
        n_lag,
        n_ft,
        n_layer,
        batch,
        epochs,
        lr,
        l1,
        l2,
        Xval= None,
        Yval= None,
        mask_value=-999.0,
        min_delta=0.0001,
        patience=10
    ):
        lstm_input = Input(shape=(n_lag, n_ft))

        # Series signal
        lstm_layer = LSTM(n_layer, activation='relu')(lstm_input)
        lstm_layer = Dropout(0.3)(lstm_layer)

        lstm_layer = LSTM((64), activation='relu')(lstm_input)
        lstm_layer = Dropout(0.2)(lstm_layer)

        lstm_layer = LSTM(32, activation='relu')(lstm_input)

        lstm_layer = LSTM(16, activation='relu')(lstm_input)

        x = Dense(n_outputs, kernel_regularizer=tf.keras.regularizers.L1L2
(l1,l2))(lstm_layer)

        self.model = Model(inputs=lstm_input, outputs=x)
        self.batch = batch
        self.epochs = epochs
        self.n_layer=n_layer
        self.lr = lr
        self.l1 = l1
        self.l2 = l2
        self.Xval = Xval
        self.Yval = Yval
        self.X = X
        self.Y = Y
        self.mask_value = mask_value
        self.min_delta = min_delta
        self.patience = patience

    def trainCallback(self):
        return EarlyStopping(monitor='loss', patience=self.patience, min_d
elta=self.min_delta)

    def train(self):
        # Getting the untrained model
        empty_model = self.model
        # Initiating the optimizer
        optimizer = keras.optimizers.Adam(learning_rate=self.lr)

        # Compiling the model
        empty_model.compile(loss='mae', optimizer=optimizer, metrics = 'ma
e')

        if (self.Xval is not None) & (self.Yval is not None):
            history = empty_model.fit(
                self.X,
                self.Y,
                epochs=self.epochs,
                batch_size=self.batch,
                validation_data=(self.Xval, self.Yval),
                shuffle=False,
                callbacks=[self.trainCallback()],
            )
        else:
            history = empty_model.fit(
                self.X,
                self.Y,
                validation_split = 0.2,
                epochs=self.epochs,
                batch_size=self.batch,
                shuffle=False,
                callbacks=[self.trainCallback()],
            )

        # Saving to original model attribute in the class
        self.model = empty_model

        # Returning the training history
        return history

    def predict(self, X):
        return self.model.predict(X)
```

Hyperparameters:

```
# Number of lags (hours back) to use for models
lag = 10

# Steps ahead to forecast
n_ahead = 4

# Share of obs in testing
test_share = 0.3

# Epochs for training
epochs = 500

# Batch size
batch_size = 64
```

```
# Learning rate
lr = 0.001

# Number of neurons in LSTM layer
n_layer = 128

# L1 Regularizer
l1=0.0001

#L2 Regulatizer
l2=0.0002

# Number of features used in the modeling:
features_final = ['HKL Features']
```

Dataset:

```
# Create Dataset:
dataset = WellH[features_final]
nrows = dataset.shape[0]

print(nrows)
```

```
1222
```

```
#Splitting Data into training/test set:

train, test = dataset[0:int(nrows * (1 - test_share))] , dataset[int(nrows
 * (1 - test_share)):]
print(train.shape, test.shape)

print(train.min(), train.max())
```

```
(855, 1) (367, 1)
HKL Features    60.0
dtype: float64 HKL Features    368.3
dtype: float64
```

```
#Data Scaling, Feature Normalization:

Min = train.min()
Max = train.max()


train =  (train-Min)/(Max-Min)
test =  (test-Min)/(Max-Min)



dataset_s = np.concatenate([train, test])

print(dataset_s.shape)
```

```
(1222, 1)
```

```
#Creating Matrix for ML Model:
X, Y = create_X_Y(dataset_s, lag=lag, n_ahead= n_ahead)

#Number of Features (Matrix Format)
n_ft = X.shape[2]
print('Number of Features:', n_ft)

#X = Number of points that the LSTM uses to make a prediction:
#Y = Number of points ahead of time that the LSTM model will predict
X.shape, Y.shape
```

```
Number of Features: 1

((1208, 10, 1), (1208, 4))
```

```
# Spliting Matrix Sequences into train and validation sets
X_train, y_train = X[0:int(X.shape[0] * (1 - test_share))], Y[0:int(X.shap
e[0] * (1 -test_share))]
x_val, y_val = X[int(X.shape[0] * (1 - test_share)):], Y[int(X.shape[0] *
(1-test_share)):]

print('Shape of Training Data:', X_train.shape)
print('Shape of Training Target Data:', y_train.shape)
print('Shape of Validation Data:', x_val.shape)
print('Shape of Validation Target Data:', y_val.shape)
```

```
Shape of Training Data: (845, 10, 1)
Shape of Training Target Data: (845, 4)
Shape of Validation Data: (363, 10, 1)
Shape of Validation Target Data: (363, 4)
```

```
# Creating the model object
model = NNMultistepModel(
    X=X_train,
    Y=y_train,
    n_outputs=n_ahead,
    n_lag=lag,
    n_ft=n_ft,
    n_layer=n_layer,
    batch=batch_size,
    epochs=epochs,
    lr=lr,
    l1 = l1,
    l2 = l2,
    Xval=x_val,
    Yval=y_val,
)
# Training the model
history = model.train()
```

```
Epoch 208/500
14/14 [==============================] - 0s 25ms/step - loss: 0.0427 - mae
: 0.0407 - val_loss: 0.0236 - val_mae: 0.0215
```

```
plt.figure(figsize=(10,8))
plt.plot(history.history['loss'], color = 'Blue', label='Training Loss')
plt.plot(history.history['val_loss'], color = 'Red', label= 'Validation Lo
ss')
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(loc='best')
plt.show()

print('Lowest MAE Obtained for training set: ' + str(min(history.history['
loss'])))
print(('Lowest MAE Obtained for validation set: ' + str(min(history.histor
y['val_loss']))))
```

```
Lowest MAE Obtained for training set: 0.04265090823173523
Lowest MAE Obtained for validation set: 0.023081675171852112
```

```
#Evaluating the Model:

# Comparing the forecasts with the actual values
yhat = [x[0] for x in model.predict(x_val)]
y = [y[0] for y in y_val]

# Creating the frame to store both predictions
seconds = WellH.index.values[-len(y):]
frame = pd.concat([
    pd.DataFrame({'time': seconds, 'HKL Features': y, 'type': 'original'})
,

    pd.DataFrame({'time': seconds, 'HKL Features': yhat, 'type': 'forecast
'})
])

# Creating the unscaled values column
frame['HKL_Features'] = [(x * (Max['HKL Features']-Min['HKL Features']) )
+ Min['HKL Features'] for x in frame['HKL Features']]

print(frame['HKL_Features'])

# Pivoting
pivoted = frame.pivot_table(index='time', columns='type')
pivoted.columns = ['_'.join(x).strip() for x in pivoted.columns.values]
pivoted['res'] = pivoted['HKL_Features_original'] - pivoted['HKL_Features_
forecast']
pivoted['res_abs'] = [abs(x) for x in pivoted['res']]
```

```
0      304.400000
1      311.400000
2      314.000000
3      313.400000
4      311.100000
          ...
358    150.384754
359    150.893009
```

```
360    152.761015
361    153.143495
362    153.034093
Name: HKL_Features, Length: 726, dtype: float64

fig, ax = plt.subplots(figsize=(20, 8))
fig.tight_layout(pad=4.0)
date_form = DateFormatter("%Y-%d-%m-%H")
ax.plot(df['Tripping_In']['2015-09-19-15':'2015-09-21-00'],
        label ='HKL Tripping In')
ax.plot(df['Tripping_Out']['2015-09-19-15':'2015-09-21-00'],
        label ='HKL Tripping Out')
ax.plot(pivoted.index, pivoted.HKL_Features_original, '-o', color='Orange'
, label='original')
ax.plot(pivoted.index, pivoted.HKL_Features_forecast, '-o',color='red', la
bel='forecast', alpha = 0.6)
ax.set_title('Original vs Predicted HKL for Well H ', fontsize=30)
ax.set_ylabel('Hookload (Tonnes)', fontsize=30)
ax.set_xlabel('Time [Year-Month-Date-Hour]', fontsize=30, labelpad = 20)
ax.xaxis.set_major_formatter(date_form)
ax.tick_params(axis='x', labelrotation=25)
ax.legend(bbox_to_anchor=(1,0.6))

plt.figure(figsize=(20,5))
fig, ax = plt.subplots(figsize=(20, 8))
fig.tight_layout(pad=4.0)
ax.plot(pivoted.index, pivoted.HKL_Features_original, '-o', color='Orange'
, label='original')
ax.plot(pivoted.index, pivoted.HKL_Features_forecast, '-o',color='red', la
bel='forecast', alpha = 0.6)
ax.set_title('Original vs Predicted HKL for Well H ', fontsize=30)
ax.set_ylabel('Hookload (Tonnes)', fontsize=30)
ax.set_xlabel('Time [Year-Month-Date-Hour]', fontsize=30, labelpad = 20)
ax.xaxis.set_major_formatter(date_form)
ax.tick_params(axis='x', labelrotation=25)
ax.legend(bbox_to_anchor=(1,0.6))
plt.show()

import random
forecast =  model.predict(x_val)

fig, axes = plt.subplots(
    nrows=4,
    ncols=2,
    figsize=(15, 15),
    facecolor="w",
    edgecolor="k"
)

indexes = random.sample(range(len(forecast)), 8 )

for i, index in enumerate(indexes):

    yhat = forecast[index]
    y = y_val[index]
    frame = pd.concat([
        pd.DataFrame({'Stand': range(len(y)), 'HKL Value': y, 'type': 'ori
ginal'}),
        pd.DataFrame({'Stand': range(len(y)), 'HKL Value': yhat, 'type': '
forecast'}),
    ])

    frame['HKL Value'] = [(x * (Max['HKL Features']-Min['HKL Features']) )
 + Min['HKL Features'] for x in frame['HKL Value']]


    sns.lineplot(x='Stand', y='HKL Value', ax = axes[i // 2, i % 2], data=
frame, hue='type', marker='o')

plt.tight_layout()

plt.show()

# Calculating the total average absolute error
error = 0
n = 0
residuals = []

for i in range(y_val.shape[0]):
    true = y_val[i]
    hat = forecast[i]
    n += len(true)

    true = np.asarray([(x * (Max['HKL Features']-Min['HKL Features']) ) +
Min['HKL Features'] for x in true])
    hat = np.asarray([(x * (Max['HKL Features']-Min['HKL Features']) ) + M
in['HKL Features'] for x in hat])

    residual = true - hat
    residuals.append(residual)

    error += np.sum([abs(x) for x in true - hat])

print(f'Final Mean Absolute Error: {round(error / n, 2)} Tonne')

Final Mean Absolute Error: 6.64 Tonne

# Flattening the list of arrays of residuals
residuals = np.asarray(residuals).flatten().tolist()
abs_residuals = [abs(x) for x in residuals]

print()

fig, ax = plt.subplots(figsize=(10,8))
sns.distplot(residuals, bins=40 ,color='Orange', axlabel = 'Residuals (Ton
ne)')
ax.set_ylabel(ylabel='Density [0,1]', labelpad = 20)
ax.set_xlabel(xlabel='Residuals (Tonne)', labelpad = 20)
ax.set_title(label='Distribution of error - Predicting 4 stands ahead',
            fontdict={'fontsize': 20})
```

```
Text(0.5, 1.0, 'Distribution of error - Predicting 4 stands ahead')
```

```
fig, ax = plt.subplots(figsize=(10,8))
ax.hist(abs_residuals, bins=40 ,color='Orange', alpha = 0.5)
# ax.set_ylabel(ylabel='Density [0,1]', labelpad = 20)
ax.set_xlabel(xlabel='Residuals (Tonne)', labelpad = 20)
ax.set_title(label='Absolute distribution of errors - Predicting 4 stands 
ahead',
            fontdict={'fontsize': 20})
```

```
pivoted.res_abs.describe()
```

```
count    363.000000
mean       5.122285
std        4.538104
min        0.030192
25%        1.898182
50%        3.794057
75%        6.885862
max       26.700251
Name: res_abs, dtype: float64
```