



Machine learning for automated stratigraphy classification

An empirical study to label subsurface formations in the Johan Sverdrup field

Petter Sølund & Mikkel Vatne Thue

Supervisors: Mario Guajardo & Julio Cesar Góez

Master thesis, Economics and Business Administration

Major: Business Analytics

NORWEGIAN SCHOOL OF ECONOMICS

This thesis was written as a part of the Master of Science in Economics and Business Administration at NHH. Please note that neither the institution nor the examiners are responsible – through the approval of this thesis – for the theories and methods used, or results and conclusions drawn in this work.

Acknowledgements

This thesis was written as a part of the Master of Science in Economics and Business Administration, with a major in Business Analytics, at the Norwegian School of Economics (NHH). This thesis constitutes 30 ECTS in our master's degree.

We would like to express our greatest appreciation to the entire Pro Well Plan team for their valuable inputs and the resources they provided. Especially thanks to Khushal Adlakha for the continuous guidance and support this past semester.

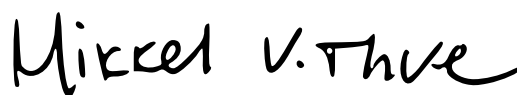
We would also like to thank our supervisors, Mario Guajardo & Julio Cesar Góez, for the fruitful discussions and their encouragement throughout the entire process.

Norwegian School of Economics

Bergen, December 2019



Petter Sølund



Mikkel Vatne Thue

Abstract

This thesis explored to what extent different supervised machine learning algorithms can be used to label subsurface formations in wells. It was explored through empirical study using wireline logs from the Johan Sverdrup field as inputs. The results from three different machine learning models were compared with the addition of a benchmark model; two LightGBM models, one LSTM model and a Logistic Regression model as a benchmark. The data set consisted of 31 wells in the Johan Sverdrup field with a total of 406 666 labeled observations and the corresponding measured properties at different depth points in the wells.

The two LightGBM models both performed better than the benchmark. The results obtained from the neural network were significantly worse than both LightGBM models and the benchmark. Due to time- and computational constraints, we were not able to fully utilize the potential of the neural network (LSTM). Hence, additional tuning and model stacking could potentially lead to improved results.

The best performing model was LightGBM 2, the model that utilized a stratified training- and validation split. Here, sequential observations from the same well were randomly split across the training- and validation data. This model yielded an accuracy of 79.17%. However, this model overfitted significantly to the training- and validation data. Further, LightGBM 1, the model that utilized a customized stratified training- and validation split, had a slightly lower accuracy of 77.58%. Here, all sequential observations from the same well were kept in the same data set, which caused significantly less overfitting to the training- and validation data. Based on this, we concluded that out of the models tested in the thesis, LightGBM 1 had the highest potential to generalize on unseen data.

The classification accuracy of around 80%, and the insight gained from the interpretable machine learning method, can be of great contribution and create significant value to experts currently performing the labeling of the formations in a manual fashion.

Keywords – Machine Learning, Interpretable Machine Learning, SHAP, LightGBM, Deep Learning, LSTM, Logistic Regression, Wireline Logs, Formation Prediction, Johan Sverdrup, Stratigraphy

Contents

1	Introduction	1
1.1	Thesis Structure	3
2	Background	4
2.1	Problem Explanation	4
2.1.1	Defining a Well	5
2.1.2	Defining Formations and Groups	6
2.1.3	Structure of a Well	6
2.2	Wireline Logging	8
2.2.1	Gamma Ray Log	9
2.2.2	Resistivity Log	10
2.2.3	Density Log	11
2.2.4	Neutron Porosity Log	11
2.2.5	Sonic Log	12
2.3	Literature Review	13
3	Methodology	16
3.1	Machine Learning	16
3.1.1	Logistic Regression	16
3.1.2	LightGBM	18
3.1.3	Recurrent Neural Network	22
3.2	Model Tuning	25
3.2.1	Tuning the Logistic Regression	26
3.2.2	Tuning the Gradient Boosting Machine	28
3.2.3	Tuning the Neural Network	29
3.3	Model Evaluation	32
3.3.1	Bias-Variance Dilemma	32
3.3.2	Train/Test Split	34
3.3.3	Performance Measures	36
3.4	Model Explanation	37
3.4.1	Additive Feature Attribution	38
4	Data Processing	42
4.1	Description of the Data Set	42
4.2	Feature Engineering	48
4.3	Data Restructuring	50
5	Analysis and Results	52
5.1	Experimental Setting 1 - Logistic Regression	52
5.1.1	Experimental Framework	52
5.1.2	Results	54
5.2	Experimental Setting 2 - LightGBM 1	56
5.2.1	Experimental Framework	56
5.2.2	Results	58
5.2.3	Model Explanation	59
5.3	Experimental Setting 3 - LightGBM 2	64

5.3.1	Experimental Framework	64
5.3.2	Results	65
5.3.3	Model Explanation	67
5.4	Experimental setting 4 - LSTM	71
5.4.1	Experimental Framework	71
5.4.2	Results	73
5.5	Summary	75
6	Discussion	79
6.1	Validity of Results	79
6.2	Domain Knowledge	80
6.3	Further Research	81
6.3.1	Other Areas and Approaches	82
6.3.2	Transfer Learning	83
6.3.3	Model Ensembling	84
6.4	The Potential of Machine Learning in Stratigraphy Classification	84
7	Conclusion	86
	References	88
	Appendix	94
A1	GitHub Repository	94
A2	Data	94
A3	Crossplot matrix	95
A4	Boxplots	97
A5	Predicted vs. Actual	99
A6	Original name and corresponding numerical values	104
A6.1	Groups	104
A6.2	Formations	105

List of Figures

2.1	Definition of a well	5
2.2	Illustration of the structure of a well	7
2.3	Illustration of a wireline tool	8
2.4	Lithologies' effect on Gamma Ray	10
3.1	Example of a decision tree	19
3.2	Level-wise vs leaf-wise growth	21
3.3	Recurrent Neural Network feedback loop	22
3.4	Structure of an LSTM node	24
3.5	Relationship between bias and variance	33
3.6	Implementation of <i>SHAP</i> for model interpretation	38
4.1	Distribution of the formations	43
4.2	Crossplot of original formations	44
4.3	Crossplot of groups	45
4.4	Crossplot of all formations	48
4.5	Input shape for LSTM networks	51
5.1	Predicted vs actual - Logistic Regression	56
5.2	Predicted vs actual - LightGBM 1	59
5.3	Accumulated feature importance for LightGBM 1	61
5.4	Feature importance LightGBM 1 - formation <i>undifferentiated nordland gp</i>	62
5.5	Feature importance LightGBM 1 - formation <i>skade fm</i>	63
5.6	Predicted vs actual - LightGBM 2	67
5.7	Accumulated feature importance for LightGBM 2	68
5.8	Feature importance LightGBM 2 - formation <i>undifferentiated nordland gp</i>	69
5.9	Feature importance LightGBM 2 - formation <i>skade fm</i>	70
5.10	Predicted vs actual - LSTM	75
5.11	Well 16/2-7 A - Predicted vs actual formation and group for all models	77
A3.1	Crossplot matrix of the formations in the data set	95
A3.2	Crossplot matrix of the groups in the data set	96
A4.1	Boxplot of the gamma ray separated by well	97
A4.2	Boxplot of the deep resistivity separated by well	98
A4.3	Boxplot of the medium resistivity separated by well	98
A5.1	Well 16/3-7 - Predicted vs actual formation and group for all models	100
A5.2	Well 16/2-14 T2 - Predicted vs actual formation and group for all models	101
A5.3	Well 16/3-6 A - Predicted vs actual formation and group for all models	102
A5.4	Well 16/2-7 A - Predicted vs actual formation and group for all models	103

List of Tables

4.1	Descriptive statistics of the data set before filter was applied	46
4.2	Filter	47
4.3	Descriptive statistics of the data set after filter was applied	47
5.1	Experimental setting 1 - Framework	53
5.2	Logistic Regression accuracy on blind wells	54
5.3	Experimental setting 2 - Framework	57
5.4	LightGBM 1 accuracy on blind wells	58
5.5	Experimental setting 3 - Framework	64
5.6	LightGBM 2 accuracy on blind wells	66
5.7	Experimental Setting 4 - Framework	72
5.8	LSTM accuracy on blind wells	73
5.9	Accuracy for all models on blind wells	76
A2.1	Missing values for variables in original data set	94
A6.1	Group dictionary	104
A6.2	Formation dictionary	105

1 Introduction

Operating in today's oil and gas industry is a challenging task. The companies' core tasks are usually performed in challenging conditions where weather and unforeseen circumstances can lead to delays and downtime. Because of the resources involved, production delays and downtime quickly becomes a huge expense for the parties involved (Wells, 2019).

Likewise, the drilling industry suffers from a complex and fragmented planning process where accessing high-quality data is the biggest bottleneck (Andresen, 2019). The companies are forced to manage more and more wells as the competitiveness and the requirement to operate at the highest potential increase. The push for profitability forces the engineers to put their energy and focus to the most profitable wells. As a result, a significant amount of wells are producing below their potential. According to Wells (2019), as little as 20% of wells are actively operated, leaving up to 80% of wells operating at a sub-optimal performance. Furthermore, the complex nature of the drilling industry also means that thousands of decisions have to be made, for instance, with respect to logistics, equipment, and staffing. Changes in the operation will, therefore, impact the safety, the performance, and the cost of the drilling process (Czypionka, Gulewicz, Keith & Rey, 2016).

In order to cope with these complex processes and reduce downtime, there has been an extensive focus on innovation and technology within the industry. Cloud computing services have made it possible to store and analyze data at a relatively low cost, which in turn has made data collection a top priority for most companies in the industry for a number of years already. Now, there seems to be a shift in priorities where companies move towards the application of methods within artificial intelligence in order to visualize and analyze available information (Crooks, 2018). This shift has a large potential for cost savings, and some estimate that better utilization of technology in order to make drilling faster and more accurate could reduce the costs for energy companies by as much as \$73 billion within five years (Nasralla, 2018).

The large consequences in terms of profitability, safety, and performance also mean that gaining a deeper understanding of the reservoir strata becomes very important in order to

improve well economics and to ensure a more accurate and efficient decision making process (Czypionka et al., 2016). Today, most companies utilize experts' domain knowledge to manually interpret geological elements, such as rock-samples directly collected from a well, in combination with different well logging measurements, in order to identify the geological structures of a well. However, collecting multiple physical samples from a well is a very expensive process, and as a result, only a few samples are normally collected per well. This forces geologists to make interpretations of the subsurface formations mostly based on other geological information, such as wireline logs and drilling penetration rates (Steingrímsson & others, 2011). Despite the fact that the industry has a lot of available data, it is lagging behind when it comes to digital-analysis. Since there are large consequences when it comes to downtime, there is reason to believe there is great potential in more data-driven decision making.

The cost of uncertainties related to the stratigraphy of a well, can be huge. Through firsthand experience and extensive market research, Magnus Tvedt (CEO of PWP), has found that if a drilling-company is given inadequate or wrongful information about the stratigraphic layers of a well, it can cause errors resulting in 1-7 days of downtime. Each day of downtime will, on average, cost 5 million NOK. Furthermore, lousy stratigraphy interpretation can lead to a well design that is sub-optimal or even result in failure. If a well design fails, the negative economic impact will result in added costs of hundreds of millions of NOK. As previously discussed, making the drilling industry faster and more accurate would not only require a lot of data, but also a well-structured database where data is accessible and ready to be analyzed. Automatic stratigraphy interpretation through wireline logs will improve and streamline a very manual task, liberating experts to do other value-creating tasks, which will further improve the well design (M. Tvedt, personal communication, December 17, 2019).

Because of the proved explanatory power of wireline logs, and sophisticated machine learning techniques' ability to identify patterns in large amounts of data, it is interesting to examine how the application of such techniques can help automate today's manual interpretation process. The main objective of this thesis will, therefore, be to explore different machine learning techniques' ability to automatically identify unique subsurface structures in wells, using wireline logs as input.

This leads to the following research question:

To what extent can machine learning techniques use wireline logs to label subsurface formations?

We will address this problem statement empirically, using well-data from the Johan Sverdrup field. All of the methods used in the thesis were implemented in Jupyter Notebooks, using the Python programming language, in Amazon Web Services (AWS) (Van Rossum & Drake Jr, 1995). The code written for this thesis can be found in the GitHub repository located in Appendix A1.

1.1 Thesis Structure

This thesis is compiled by 7 chapters. Chapter 2 gives a detailed explanation of the problem at hand and introduces relevant background information about the structure of a well, stratigraphic formations, and wireline logs. Chapter 3 presents the methodological framework of the algorithms used to address our problem statement and describes how these are implemented and evaluated. Chapter 4 describes the data and how it was preprocessed before it was introduced to the models. In chapter 5, four different experimental approaches are analyzed, and the result is presented. Chapter 6 presents a discussion with respect to the limitations and validity of the results, together with suggestions for further research. Finally, Chapter 7 summarizes and concludes this thesis.

2 Background

This chapter serves as an essential introduction to the fundamental concepts and techniques relevant to our problem statement. The first section discusses the problem at hand and describes how a well in our data set is structured. The second section gives a brief overview of the different logs used to classify formations. Lastly, an overview of relevant scientific literature is discussed.

2.1 Problem Explanation

As previously introduced, the main objective of this thesis is to study how different machine learning techniques can use wireline logs to label subsurface formations. By combining state of the art machine learning techniques' ability to learn from big data, and the proved explanatory power of well logs with respect to subsurface structures, there is reason to believe that automation of the classification process can create significant value for all parties involved.

By Norwegian law, companies are obligated to share recorded drilling logs to the Norwegian Petroleum Department (NPD) (Directorate, 2019). From these publicly available sources, our collaborating partner, Pro Well Plan (PWP), has gathered and pre-processed more than 2000 data sets on unique wells across the Norwegian Continental Shelf. About 600 of these well data sets are labeled in great detail with rock-formations, by professionals from various companies in the industry. We saw it as a great opportunity to utilize the availability of this data to answer our problem statement.

Together with PWP, we decided to address this problem empirically, using data from the Johan Sverdrup field. The Johan Sverdrup field was chosen because it is a new and promising field for the future of oil production at the Norwegian Continental Shelf. The field was discovered in 2010, and it is the third-largest Norwegian discovery of all time. The first production phase was started in October 2019. The field is expected to continue its production for more than 50 years, and will at its peak, account for more than 30% of the total oil production in Norway (Equinor, 2019). Furthermore, because the Johan Sverdrup field was discovered so recently, the data collection is performed using more

advanced equipment, which is likely to yield data of higher quality compared to data recorded for older wells. This is important since good data quality will be crucial for the performance of the machine learning techniques utilized in the thesis.

2.1.1 Defining a Well

The question at hand deals with two different parts that both originate from wells; the subsurface formations and the wireline logs, also known as well logs. It is, therefore, important to understand what formations are how these are structured in a well. The definition of a well is based on the Norwegian Petroleum Directorate definition of a well: "a borehole which is drilled in order to discover or delimit a petroleum deposit and/ or to produce petroleum or water for injection purposes, to inject gas, water or other medium, or to map or monitor well parameters" (Norwegian Petroleum Directorate, nda). However, as illustrated in Figure 2.1, a borehole can have one or more wellbores, also known as well paths. Since our data set consists of individual well paths, a well in this thesis is defined as the individual wellbores that have been drilled.

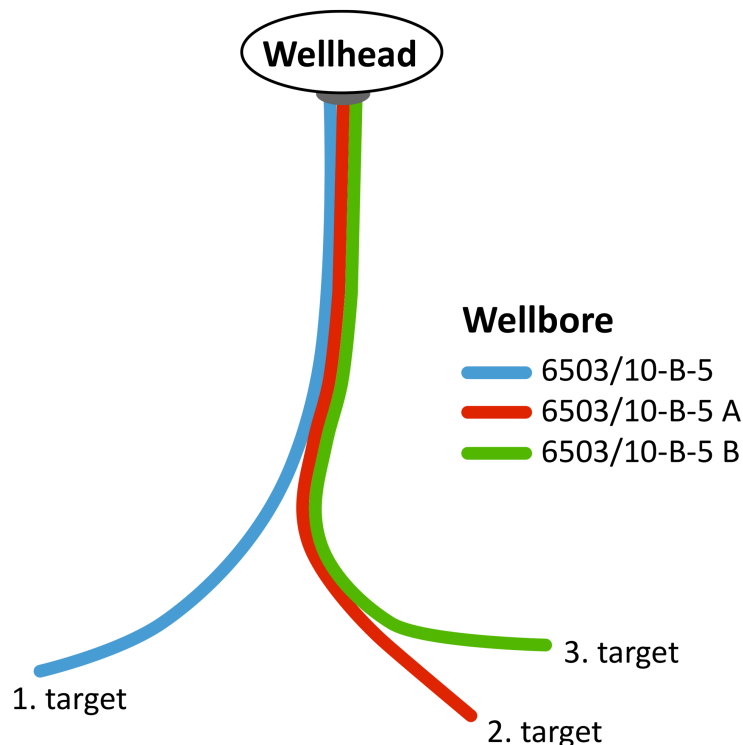


Figure 2.1: A well in this thesis is defined as the individual wellbores, identified by the colors blue, red, and green in the figure. Reprinted from "Well classification" by Norwegian Petroleum Directorate (ndb).

2.1.2 Defining Formations and Groups

In addition to the definition of a well, it is also crucial to understand what a formation is and how it can be distinguished from groups. Both formations and groups can be described as lithostratigraphic units that are classified based on their physical and mineralogical characteristics, in addition to their relationship with surrounding rocks. A lithostratigraphic unit is defined based on its composition of the three main types of rock, sedimentary, igneous, or metamorphic-equivalent. The most important take-away from this is the principle that younger layers are established over older layers, meaning the sequence of the formations matters as the younger layers will depend on the older layers and vice versa (Geological Survey of Norway, 2015). Lithostratigraphic units can be separated into formations and groups. A formation is the primary formal unit of lithostratigraphic classification, while a group refers to a succession of two or more contiguous or associated formations with significant and diagnostic lithologic properties in common (Salvador & Murphy, 1998).

2.1.3 Structure of a Well

During the drilling of the wells, different measurements are collected in order to analyze the strata further. The measurements can be seen in conjunction with, for instance, the formations in the well. An example of such can be seen in Figure 2.2. In the figure, we have plotted six wireline logs from well *16/5-2 S*, one of the wells in our data set, against the true vertical depth of the well on the y-axis. The figure illustrates how the wells in our data set are structured and how each well can be viewed as a subsection of the subsurface strata. By plotting the different wireline logs against the true vertical depth, it is possible to visualize the changes in the logs, depending on the depth point in the well. Furthermore, Figure 2.2 also shows the formations and the corresponding group at each depth point. Each color in the "Formations"-plot represents a unique formation, and the same goes for each color in the "Group"-plot. As can be seen in the figure, the formations in the well vary depending on the depth of the well and the values for the different wireline logs.

In Figure 2.2, we have highlighted a subsection in order to illustrate the main objective of

the thesis clearly. As can be seen in the highlight, the values of gr , $rdep$, $rmed$ and $nphi$ change as the formation changes. In this case, there is a significant drop in the values of the above-mentioned logs; however, in some cases, the change might be more subtle. The goal of the thesis is to investigate the extent to which machine learning models are able to learn the patterns and other information from the wireline logs, in order to correctly label the formations.

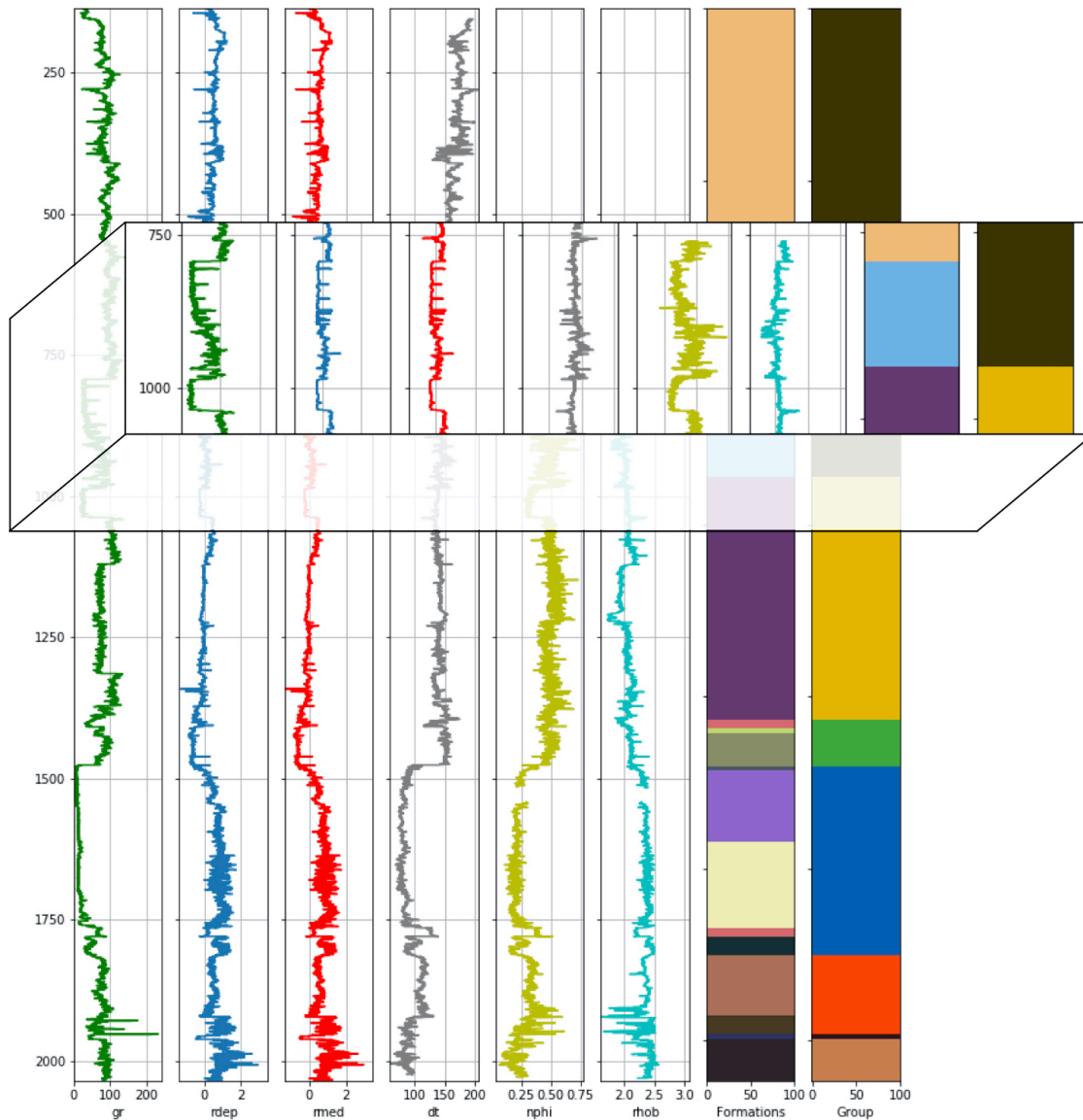


Figure 2.2: Illustration of the structure of a well. The figure illustrates the values of the wireline logs in well $16/5-2 S$ at each depth point and the corresponding formation and group. Each color in the formation plot corresponds to a unique formation in the data set. This is also true for the groups. The highlighted subsection of the figure shows how the values of the wireline logs change when the formation and group changes.

If we compare the formations to the groups, Figure 2.2 further shows how formations close

to each other often have similar properties, placing them in the same "parent"-group.

Furthermore, proper domain knowledge is highly beneficial when working with machine learning. Knowledge about the input data can, for instance, help when determining the data processing pipeline. In this thesis, the input data is wireline logs, and it is therefore important to have some knowledge of the process behind the information extraction and the information contained in the logs.

2.2 Wireline Logging

To get accurate information about the structure of the lithostratigraphic units in a well, extracting core samples of reservoir-rock would give the most accurate results. However, this is a very comprehensive and expensive process, and therefore not common practice in the industry today (Dubois, Bohling & Chakrabarti, 2007). With the high number of core samples required to get sufficient information about a well, other methods are crucial in order to interpret ground stratification characteristics effectively.

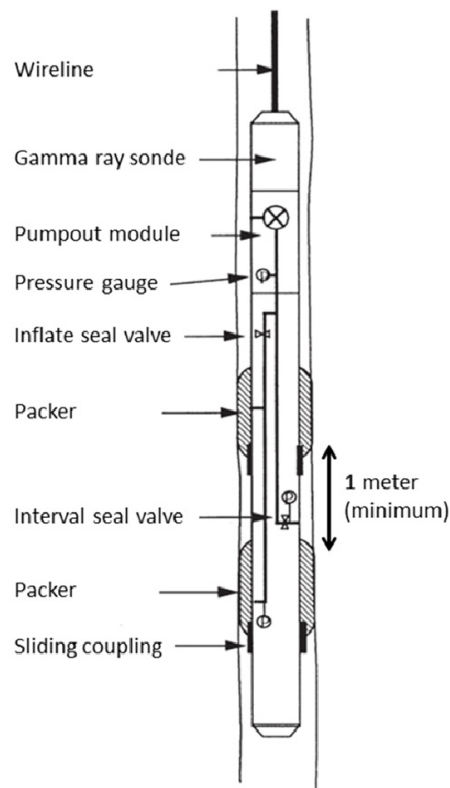


Figure 2.3: Illustration of a wireline tool. Reprinted from "Feasibility and Design of Hydraulic Fracturing Stress Tests Using a Quantitative Risk Assessment and Control Approach" by Bérard et al. (2019).

The most common method in today's industry is wireline logging or downhole logging. Wireline logging can be described as a process of recording detailed information about geological formations, plotted against each depth-point of the well (Leyland, 2017). This information is retrieved by lowering a wire, equipped with different types of measuring tools, down the borehole. The various measuring tools then record different signals, such as resistivity, density, or porosity, at each given depth-point of the well. This can be done in between drilling operations or at the end of drilling (Vakarelov, 2016).

2.2.1 Gamma Ray Log

A gamma ray log, or natural radiation log, contains the measured radiation decay from rocks in subsurface formations. The logging tool measures the total gamma ray activity, which is a sum of isotopes emanating from three different chemical elements; Uranium, Thorium, and Potassium (Olejnik, Karaffa & Fleming, nd). The isotopes released from these chemical elements can form nuclear energy-level structures, which lets us identify and measure the presence of radioactive elements through the release of gamma ray radiation (Stark, 2018).

When isotopes are liberated from rocks in the formations, the released energy will be reduced consecutively as atoms collide in the rock-formation. This process is called "Compton Scattering" and will continue to occur until the energy level is so low that it is absorbed by the rock-formation. In other words, the measured gamma ray value will depend on how powerful the initial release radioactive elements were and how much Compton Scattering the radioactive elements have encountered. The level of Compton Scattering will depend on the density of the formation (Glover, 2014).

Gamma ray is a useful tool when it comes to lithology interpretation. Even if the gamma ray log alone does not give us enough information to define unique lithology-formations, it becomes valuable when combined with other logs (Glover, 2014).

It is also important to note that the reliability of the gamma ray measure will depend on the borehole quality. Since the gamma ray measuring tool is typically going through the center of the borehole, intervals of measured values can be affected if the hole suffers from caving. An example of caving can be seen at the top of Figure 2.4. When caving

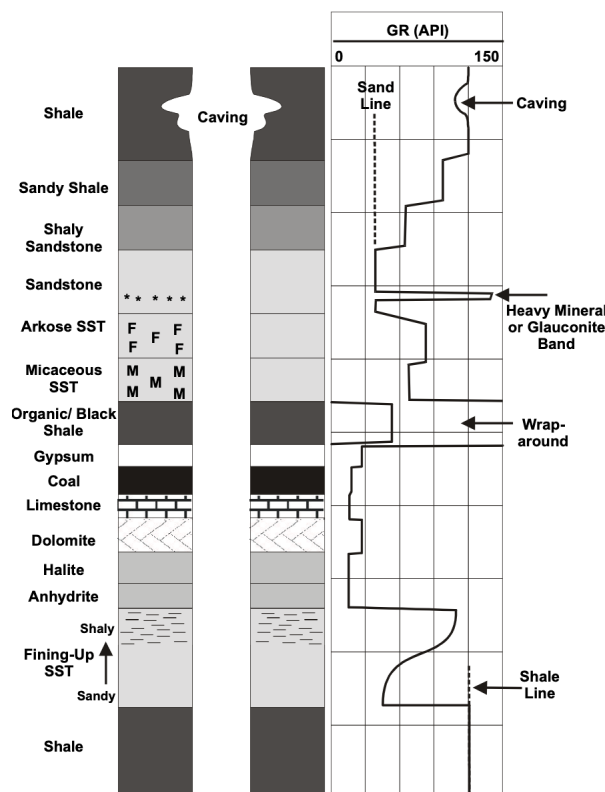


Figure 2.4: Graphic overview of the effect different types of lithologies have on Gamma Ray. Reprinted from "Petrophysics MSc Course Notes" by Glover (2014). The figure illustrates how different types of rocks can be separated by different levels of gamma ray. It also shows how the borehole width and caving can affect the reliability of measures.

occurs, we will have more drilling mud between the formation and gamma ray and with that reduce the radioactive signal produced by the formation. Higher mud density will give greater underestimation (Glover, 2014).

2.2.2 Resistivity Log

Resistivity logs, or electrical logs, measures the resistivity of subsurface formations through electrical signals. There are three main types of resistivity measures, shallow-, medium-, and deep resistivity. The resistivity tool identifies hydrocarbon-bearing versus water-bearing zones, as this can help us indicate permeable zones, and to determine rock-porosity (Asquith, Krygowski, Henderson & Hurley, 2004). A rock is composed of thousands or millions of grains, and the density of these grains will depend on the porosity of the given rock. If a rock has high porosity, the space between the grains is bigger, which makes the rock more permeable. A permeable rock can carry more fluids, such as water and

oil. A rock's grains are non-conductive, or in other words, they are not able to transfer electricity, making them highly resistive. On the contrary, water will be highly conductive, and with that have very low resistivity.

Because the solid materials are non-conductive, a rock's ability to transfer electricity will almost explicitly be a function of the fluids in the pores (Rider, 2011). Hydrocarbons (petroleum) have higher resistivity compared to saltwater, and saltwater has higher resistivity compared to freshwater. As a result, resistivity measures can help separate different types of formations. As an example, sandstone formations will have higher resistivity than shale formations, because shale formations have more water in the clay (Asquith et al., 2004).

2.2.3 Density Log

A density log is a record of a formation's bulk density. The bulk density is the overall density of a rock, which includes both the solid matrix and the fluid that is enclosed in the rock. The log can be used to calculate the porosity and the hydrocarbon density, but its most important use case is in the identification of certain minerals (Rider, 2011).

The tool subjects the formations to medium-energy gamma rays in order to measure the attenuation between the tool source and the detectors. The tool is, in reality, measuring the electron density, which in turn is closely related to the common density of the formations. Research has shown that the density tool's depth of investigation is very shallow, most likely around 10cm for average densities, meaning the tool is profoundly affected by hole conditions (Rider, 1991).

2.2.4 Neutron Porosity Log

The neutron log is a record of how a formation reacts to neutron bombardment. Since formations with high water content absorb neutrons rapidly, the log is in principal a measure of the water content in the formation. The log is used to measure the porosity of a formation, and this works very well as a discriminator between oil and gas (Rider, 2011). The neutron porosity log is often combined with the density log on compatible scales,

and the combination of the two logs is one of the best subsurface lithology indicators available. This is due to the fact that both the neutron log and the density log both measure the porosity of a formation, and differences between the two logs can be useful when identifying certain formations (Rider, 1991).

The tool consists of a neutron source and two detectors where the detectors register the degradation in energy of the neutrons radiated from the source. The reason the neutron log tool works so well in identifying the water content of the formations is the fact that neutrons have no electrical charge, and the mass is equivalent to that of a hydrogen nucleus. That means that the neutrons will lose energy when they collide with hydrogen nuclei, whereas colliding with nuclei with heavier or lighter mass will not result in significant energy loss (Rider, 1991).

2.2.5 Sonic Log

A sonic log, also known as an acoustic log, indicates the time it takes for high-frequency pulses to travel through formations that are close to the borehole. The sonic log tool consists of an acoustic transmitter and two receivers, spaced at different distances from the transmitter, and the tool is mostly run hole-centered. This is done in order for the sonic pulse to be radiated symmetrically and so that the measurements come from all sides of the hole simultaneously. Since the formations have different capacities when it comes to the transmission of sound waves, the log can be used to evaluate the porosity in liquid-filled holes (Rider, 1991).

The measurements from the sonic tool are often cross-multiplied with the measurements from the density log in order to make the acoustic impedance log. Qualitatively a geologist can use the sonic log to identify source rocks, overpressure, and to some extent, fractures (Rider, 2011).

It is important to note that the measurements from the sonic log tool can be affected by poor boreholes. An example of a poor hole is caving, as was seen in Figure 2.4. Since prolonged exposure to drilling muds can cause deterioration, the sonic logs that are registered soon after logging are the most reliable measurements (Rider, 1991).

2.3 Literature Review

This section will give a brief overview of the available research related to the problem statement, while the literature in regards to the models will be discussed in Section 3.

Machine Learning and Stratigraphy

In recent years, machine learning has successfully been implemented in order to solve numerous different problems. In particular, deep learning has recently proven to perform classification tasks at a human level, in what can be described as an AI-revolution (Chollet, 2018).

An example of what to expect can be observed in the promising results that artificial neural networks (ANN) have shown on reservoir data. Ayala & Ertekin (2005) proposed the implementation of an ANN in gas-condensate reservoirs, and their tool is capable of assisting the engineers when they are designing an optimized exploration scheme for a particular reservoir under consideration for development. Another method has been developed using an ANN linked to the particle swarm optimization (PSO) tool, where the goal is to forecast the productivity at the initial stages of the development of horizontal well drilling (Ahmadi, Soleimani, Lee, Kashiwao & Bahadori, 2015).

Further, Wang, Yang, Zhao & Wang (2018) used a data mining and a machine learning approach in order to classify the reservoir pore structure in the Mesozoic strata. Their technique classified the pore structure into four types with a cross-validation accuracy of 75%. In addition, Zazoun (2013) developed an ANN in order to predict fracture density from conventional well logs. Their study demonstrated a good agreement between the neural network model prediction and core fracture measurements, indicating that the inputs in conventional well logs are suitable inputs in a neural network model.

Machine Learning Contest

Machine learning has also been implemented in order to classify the facies, in oil wells, using well-logs as input. This application gained increased popularity through the SEG machine learning contest, launched by Brandon Hall in 2016 (Hall, 2016). In this competition, a

data set consisting of nine wells from the Hugoton and Panoma fields in southwest Kansas and northwest Oklahoma was provided, and the goal was to achieve the highest facies-classification accuracy. During the competition, more than 300 entries were submitted by 40 teams, and the top five contributions achieved an accuracy of 0.62 - 0.6388 on two blind wells. All top five contestants used an XGBoost model in Python (Hall & Hall, 2017).

The above-mentioned data set was also the basis for some attempts at using deep learning models to predict the facies as discussed by Hall (2019). One of the deep learning models was a one-dimensional convolutional neural network (CNN) proposed by Imamverdiyev & Sukhostat (2019). The 1D-CNN model showed statistically significant improvements and outperformed many of the models they used as a comparison. Imamverdiyev & Sukhostat (2019) conclude that their deep learning model can be useful for future research and facies identification. Lasscock (2019) also experimented with deep learning through the use of an LSTM-model, treating the input as a time series problem.

In summary, research shows that machine learning has proved to be a useful tool in the oil and gas industry and that well logs are considered useful inputs when predicting facies in oil wells. However, it is important to keep in mind that the stratification, the layering that occurs in most sedimentary rocks, is very different in different areas of the world (of Encyclopaedia Britannica, 2014). This means that the measurements found in the well logs are different, and without testing, there is no guarantee that the models used to classify facies in other areas of the world will perform just as well on strata in a different area.

Implementations on the Norwegian Continental Shelf

Since most of the previously discussed research is based on strata in other parts of the world than Norway, it is interesting to see how certain machine learning models will perform on a data set with wells from the Norwegian Continental Shelf. When it comes to facies predictions in Norway, there have been some attempts at classifying facies through machine learning, where the first attempt came as early as 1992. Bølviken, Storvik, Nilsen, Siring & Van Der Wel (1992) addressed in their paper whether a computer can be programmed to identify depositional facies from a set of wireline logs through what is

now known as supervised learning. The conclusion was that the results of the tests are promising, and as new algorithms have been developed, it would be relevant to see how well the new models will perform. The most recent attempt using machine learning to classify facies through wireline logs in Norway came from Bøe (2018). In his master's thesis, he used an XGBoost model on input data from an area on the Norwegian Continental Shelf. The thesis gives a good background for the use of machine learning on wireline logs. However, we note that in his thesis, the models was compared using Root Mean Square Error (RMSE) on a classification task. Therefore, we are not able to draw conclusions from the machine learning section of his thesis.

To the best of our knowledge, there have recently been no attempts at using machine learning to classify the formations in a well using wireline logs as input, on the Norwegian Continental Shelf. Further, most of the attempts have used traditional machine learning only, and with the promising result that deep learning has shown lately, classification of formations through the use of deep learning models is definitely an area worth researching.

It is also worth noting that all the above-mentioned research that included classification tasks in wells aimed at classifying the facies. Formations are different from facies in the sense that a formation corresponds to a body of rock that is distinguishable from the rock above and below in the stratigraphic sequence, while facies is a set of rocks that were deposited in the same sedimentary environment (Rey, Galeotti & others, 2008). As far as we know then, there has been no research on formation classifications using machine learning and, in particular, deep learning, with wireline logs as input.

3 Methodology

This chapter is divided into three parts, where the first section presents the theoretical framework for the algorithms used in our thesis. The second part of the chapter relates to how the algorithms have been implemented. The third section describes how the models were evaluated. Lastly, the theoretical framework of the interpretable machine learning method SHAP is introduced.

3.1 Machine Learning

3.1.1 Logistic Regression

The Logistic Regression model was used as a benchmark in order to evaluate if additional effort, through feature engineering and construction of more complex and computationally expensive models, would be worthwhile with respect to both resources and increased performance. The model is a technique from the field of statistics and is by many considered the go-to method for classification problems with a binary outcome (Géron, 2019). The Logistic Regression model is based on the Logistic Function, which can be seen in Equation 3.1 and the output of the function is the probability for all values of X (James, Witten, Hastie & Tibshirani, 2013).

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} \quad (3.1)$$

Through manipulation and by taking the logarithm of both sides, we are given Equation 3.2. The left-hand side of the equation is known as the *log-odds* or *logit* and the equation shows that the Logistic Regression model in Equation 3.1 has a *logit* that is linear in X (James et al., 2013). It is important to note that the Logistic Regression model is different from a Linear Regression model in the sense that a one-unit increase in X changes the log odds by β_1 . However, since the relationship between $p(X)$ and X is not linear, as seen in Equation 3.1, β_1 does not correspond to the change in $p(X)$ associated with a one-unit

increase in X , but the change in $p(X)$ would depend on the current value of X (James et al., 2013).

$$\log \left(\frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X \quad (3.2)$$

In order to calculate the coefficients in the logistic function, the *maximum likelihood* method is preferred. The intuition behind is that we aim to find estimates for β_1 and β_0 such that the predicted probability $\hat{p}(x_i)$, using Equation 3.1, corresponds to the observed probability in the data set (James et al., 2013). The mathematical equation for the likelihood function can be seen in Equation 3.3.

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=1} (1 - p(x_{i'})) \quad (3.3)$$

To generalize the Logistic Regression model so that it supports multiple classes, the Logistic Regression model is combined with a Softmax Regression model. The Softmax Regression model first computes a score $s_k(X)$ for each class k , using Equation 3.4 (Géron, 2019).

$$s_k(X) = X^T \theta^{(k)} \quad (3.4)$$

When the score of each class k for the instance X is calculated, the probability \hat{p}_k for each class k can be estimated, using Equation 3.5, where K is the number of classes, $s(X)$ is a vector containing the scores of each class for the instance X and $\sigma(s(X))_k$ is the estimated probability that the instance X belongs to class k , given the scores of each class for that instance (Géron, 2019).

$$\hat{p}_k = \sigma(s(X))_k = \frac{e^{(s_k(X))}}{\sum_{j=1}^K e^{(s_j(X))}} \quad (3.5)$$

When the probability \hat{p}_k that the observation X belongs to class k has been calculated, the regression classifier simply predicts the class with the highest estimated probability (Géron, 2019).

3.1.2 LightGBM

This section serves as a short overview of decision trees and boosting algorithms. Furthermore, since the Light Gradient Boosting Machine (LightGBM) algorithm is built on a boosted decision tree algorithm, the LightGBM paragraph of this section does not go into further details in regards to decision trees and boosting, but focuses instead on aspects of LightGBM that may differ from other boosting algorithms.

Decision Trees

Tree-based prediction methods are based on the segmentation of the predictor space into a number of simpler regions as exemplified in Figure 3.1 (James et al., 2013). The regions of the predictor space are known as terminal nodes or leaves and the segments of the trees that connect the internal nodes are referred to as branches. In Figure 3.1 the leaves are the predictions 5.11, 6.00, and 6.74, while the internal nodes are indicated by the text "Years<5.5" and "Hits<117.5" (James et al., 2013).

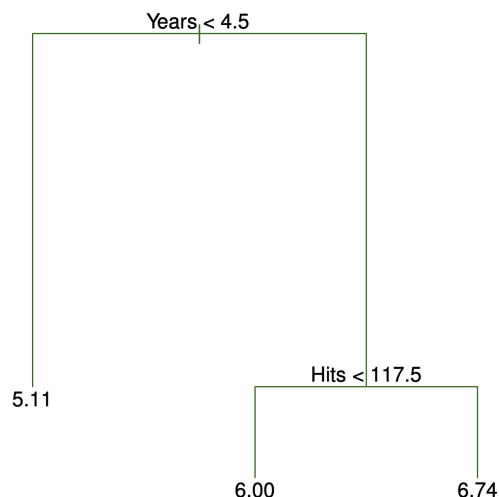


Figure 3.1: Example of a decision tree. Reprinted from "An Introduction to Statistical Learning" by James et al. (2013). The figure is an example of a regression tree in which the predicted salary of a baseball player is a function of the number of years the player has played and the number of hits he has made in the previous year. If the player has played less than 4.5 years in the major leagues the predicted salary is 5.11. Contrary, if the player has played 4.5 years or more in the major leagues his predicted salary will depend on whether or not his number of hits were above or below 117.5.

Algorithm 1 Building a regression tree. Reprinted from "An Introduction to Statistical Learning" by James et al. (2013)

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
 2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α .
 3. Use K-fold cross-validation to choose α . That is, divide the training observations into K folds. For each $k = 1, \dots, K$:
 - (a) Repeat Steps 1 and 2 on all but the k th fold of the training data.
 - (b) Evaluate the mean squared prediction error on the data in the left-out k th fold, as a function α .Average the results for each value of α , and pick α to minimize the average error.
 4. Return the subtree from Step 2 that corresponds to the chosen value of α .
-

Algorithm 2 exemplifies how a regression tree is built and can be seen in conjunction with Figure 3.1. Both Algorithm 2 and Figure 3.1 are examples of a regression tree, but we note that a classification tree works in the exact same way as the regression tree described except that instead of predicting a quantitative response, the prediction is a qualitative one. The effect of this is present in the way the predicted response of an observation is given. In a regression tree the predicted response of an observation is given by the mean response of the observations that belong to the same terminal node.

A classification tree on the other hand, predicts the observation to belong to the most commonly occurring class of training observations in the region to which it belongs (James et al., 2013). Despite some of the advantages of a decision tree; it is easily interpreted, and can handle qualitative predictors well, by itself a decision tree generally has a lower predictive performance than some of the other machine learning models. However, the aggregation of decision trees, through for instance boosting, can significantly improve the prediction power of a decision tree (James et al., 2013).

Boosting

Boosting is an approach for improving the prediction results from a decision tree. In boosting, several trees are grown sequentially, meaning that each tree in the model is grown using information from the previously grown trees (James et al., 2013). With this approach a new tree is grown from the residuals rather than the outcome of the model. The decision tree that was grown from the residuals is then added to the fitted function in order to update the residuals. Lastly, the final prediction is given by the weighted average of all sequentially grown predictors (James et al., 2013). Mathematically boosting can be expressed using Algorithm 2.

Algorithm 2 Boosting for Regression Trees. Reprinted from "An Introduction to Statistical Learning" by James et al. (2013)

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set
2. For $b = 1, 2, \dots, B$ repeat:
 - (a) Fit a tree \hat{f}^b with d splits ($d + 1$ terminal nodes) to the training data (X, r)
 - (b) Update \hat{f} by adding in a shrunk version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x) \quad (3.6)$$

- (c) Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i) \quad (3.7)$$

3. Output the boosted model,

$$\hat{f}^b(x) = \sum_B^{b=1} \lambda \hat{f}^b(x) \quad (3.8)$$

LightGBM

Boosted trees have recently grown in popularity due to its efficiency, accuracy, and interpretability. However, in recent years boosted trees have faced challenges due to the emergence of big data. A conventional boosted tree scans all the data instances in order to estimate the information gain of all the possible split points, for every feature in the data set. As a result, handling big data using boosted trees is very time consuming (Ke, Meng, Finley, Wang, Chen, Ma, Ye & Liu, 2017).

To resolve this issue, LightGBM was created. The LightGBM algorithm is different from traditional boosted trees in the sense as includes two new techniques; Gradient-based One-Side Sampling (GOSS), and Exclusive Feature Bundling (EFB). The result is an algorithm that can accelerate the training process by more than 20 times while achieving almost the same accuracy (Ke et al., 2017).

The idea behind *GOSS* is that the instances with larger gradients will contribute more to the information gain, meaning that when down sampling the data instances it is important to keep the instances with large gradients. The small gradients will then randomly be dropped and the result is a treatment that can lead to a more accurate gain estimation than uniformly random sampling, with the same target sampling rate (Ke et al., 2017).

EFB is based off the idea that in real applications, although there are a large number of features, the feature space is quite sparse. It is therefore possible to reduce the number of effective features without losing important information by the use of a bundling algorithm (Ke et al., 2017).

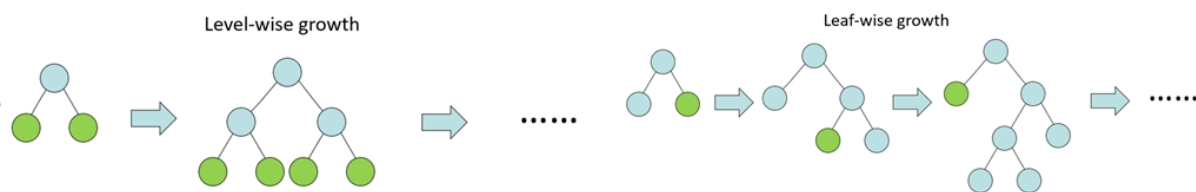


Figure 3.2: Level-wise vs leaf-wise growth. Reprinted from "LightGBM and XGBoost Explained" by Kurita (2018).

Some of the other ways in which LightGBM is different from traditional tree-based models is the use of a *leaf-wise tree growth* approach instead of a *level-wise tree growth* approach, as can be seen in Figure 3.2. The level-wise growth approach is less prone to overfitting,

however, since it is less flexible, the leaf-wise growth approach tend to achieve lower loss and the flexibility makes it a suitable choice for large data sets (Shi, 2007). Furthermore, LightGBM uses histogram-based algorithms instead of pre-sort based algorithms. The utilization of histogram-based algorithms leads to reduced cost of calculating the gain for each split since the algorithm will bucket features with continuous values into discrete bins and then use the bins to construct feature histograms during training. Once the histogram has been constructed, the time complexity will be based off the number of bins which will be smaller than the full data set (Ke et al., 2017).

3.1.3 Recurrent Neural Network

In our thesis, we have utilized a Recurrent Neural Network (RNN) in order to make predictions. As is illustrated in Figure 3.3, the Recurrent Neural Network is different from a feedforward network. In particular, an RNN iterates through the sequence elements while simultaneously maintaining a state that contains information relative to the previous information (Chollet & Allaire, 2018). In other words, where as a feedforward network only considers the current input, the looping mechanism in an RNN allows it to take into consideration the previous inputs as well as the current input (Donges, 2018). Due this, RNNs are better equipped to detect patterns in the input sequence and are therefore often used for time series forecasting (Weller, 2018).

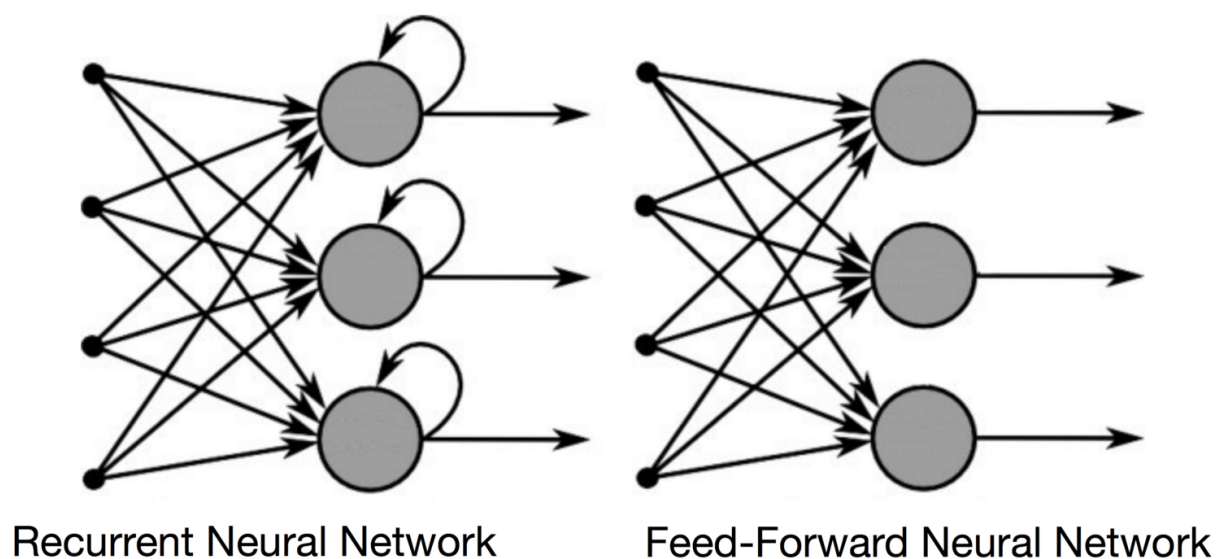


Figure 3.3: Recurrent Neural Network feedback loop. Reprinted from "A Survey on the Application of Recurrent Neural Networks to Statistical" by De Mulder et al. (2014).

A downside of RNNs is the computational power required to run the algorithms. The fact that they take both past and present inputs are taken into consideration means they compared to other algorithms they are very computational expensive. Another problem more prevalent in RNNs compared to other algorithms is the problem of vanishing/exploding gradients. Vanishing/exploding gradients occurs when the gradient of the network ends up with many similar numbers. If the gradient values then become extremely small or extremely large they are not able to contribute to- or they result in very large updates to the weights of the model (Brownlee, 2017a). If this is the case information from the earlier steps might be lost as the model is not able to carry the information forward. The fact that this is such a common problem in RNNs, the networks are often described as networks with short-term memory. To resolve this issue a modification of the traditional RNN was created, Long Short-Term Memory (LSTM).

LSTM

Long Short-Term Memory networks were introduced by Hochreiter & Schmidhuber (1997) and their solution to the short-term memory problem is the use of additional neural networks. The addition networks regulate the flow of information through the sequence chain which allows past information to be reinjected at a later time (Nguyen, 2018). The neural networks within the network itself are referred to as gates and in an LSTM network there are three gates in total. These are the *forget gate*, the *input gate* and the *output gate*. Together they have the ability to remove or add information to the cell state. This means that each LSTM unit makes decisions by considering the current input, previous output and previous memory (Yan, 2016). Figure 3.4 shows a detailed explanation of the repeating module in an LSTM network and will, in combination with Equation 3.9, serve as a brief overview of the network.

The first sigmoid activation function from the left in the illustration is what is called the "forget gate layer", f_t . In the forget gate, the output of the previous block, h_{t-1} , multiplied by the weight, W_f , is concatenated with the current input, x_t , multiplied by the weight U_f and the bias of the forget gate, b_f , is added. Next, a sigmoid activation function is applied, making sure the output is compressed between 0 and 1. An activation function can in many ways be seen as a valve and determines how much and what information

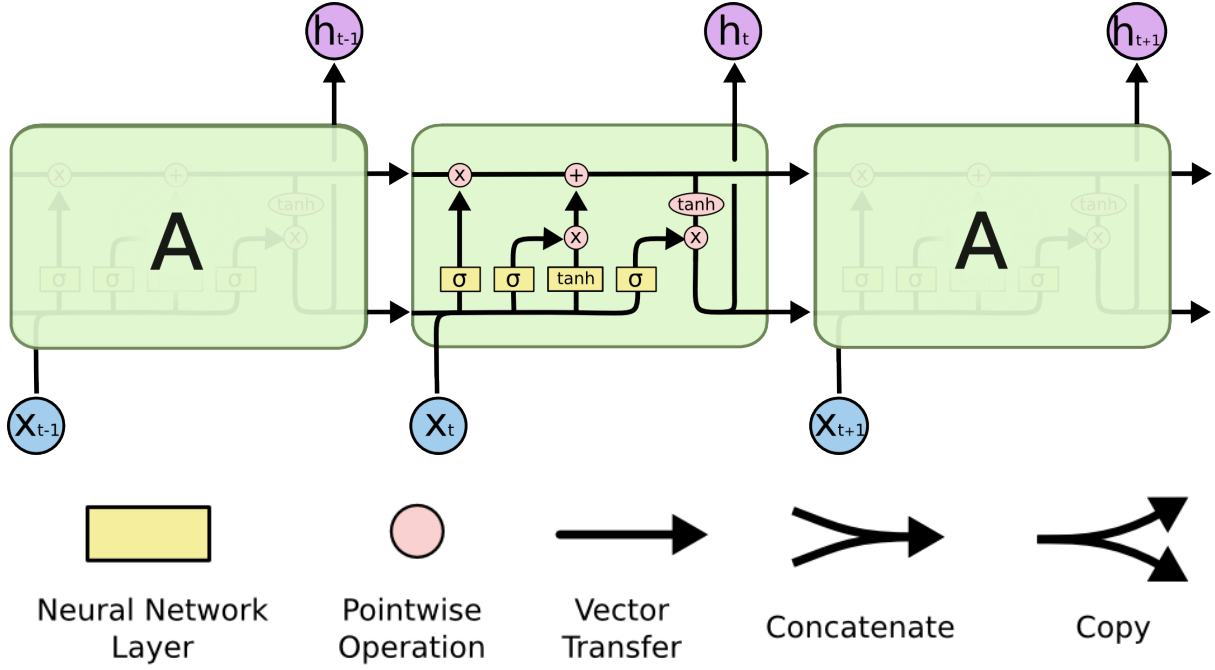


Figure 3.4: Structure of an LSTM node. Reprinted from "Understanding LSTM Networks" by Olah (2015).

that should be passed on to the next layer (V, 2017). As can be seen in 3.9, the same structure applies to both the input gate, i_t , and the output gate, o_t . However, the bias and weights are individual for each of the gates.

$$\begin{aligned}
 f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
 i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
 \tilde{C}_t &= \tanh(W_C x_t + U_C h_{t-1} + b_C) \\
 C_t &= f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \\
 o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
 h_t &= o_t * \tanh(C_t)
 \end{aligned} \tag{3.9}$$

In addition to the previously mentioned gates, LSTM has a cell state, C_t . To calculate the cell state, the Hadamard product of the forget gate, f_t , and the previous cell state, C_{t-1} , is added element-wise to the Hadamard product of the input gate, i_t , and \tilde{C}_t . Here, \tilde{C}_t is a vector of new candidate values that could be added to the state.

Lastly, the output vector, h_t is calculated as the Hadamard product of the output gate, o_t

and the current cell state C_t after a *tanh* activation function has been applied. The *tanh* activation function makes sure that the values are between -1 and 1.

To summarize, by combining all the elements in 3.9, the LSTM network is able to use past information in addition to current information when making predictions. The forget gate is a neural network that decides what information from the cell state that should be thrown away, while the input gate decides what new information that will be stored in the cell state. This consists of a layer that decides which values to update in addition to a layer that creates a vector of new candidate values that potentially will be added to the cell state. The output gate in combination with the a filtered cell state determines what prediction that are being made (Olah, 2015).

3.2 Model Tuning

For machine learning, one can generally say that finding the optimal combination of tuning parameters is a very time-consuming and challenging task. By applying machine learning techniques, such as LightGBM and deep learning, hundreds of decisions need to be made. All of which will have a significant impact on the model outcome. Examples of such parameter-decisions can be the number of boosted trees to pick, what loss function the model should try to minimize, and so on. If we make inadequate or wrongful decisions with respect to parameters, we might experience things like overfitting to our training data, or that the model will not be able to learn at all. We can divide parameters into two separate categories; consistent parameters and hyperparameters. This section discusses the approaches taken in order to tune the machine learning models. Since the model tuning is such an important part of our thesis and algorithms have different, and intricate tuning parameter, the section is divided into subsections for each individual model.

Consistent parameters

Some parameters remained the same throughout the whole training process. Consistent parameters are usually problem specific and must be chosen with intuition based on what you are trying to predict. For instance, it is essential to choose a loss function that reflects the multi-class problem at hand. If we were to choose a loss function suited for a

regression problem, the model would be penalized more by predicting class 10 compared to predicting class 2 if the target class was 1. In a regular classification problem, there is not a numerical relationship between the classes, hence predicting class 10 and 2 would be equally wrong.

Hyperparameters

Hyperparameters, also known as tuning parameters, are parameters that control the models' training process and have a critical effect on the models' ability to learn. As machine learning techniques get more sophisticated, the number of hyperparameters to tune is rapidly increasing, making it harder and harder to tune machine learning models manually. In other words, model tuning has become a very challenging and time-consuming task, even for researchers with an extensive machine learning experience and domain knowledge. Although it is possible to gain some tuning-skills, manual tuning is likely to yield sub-optimal results at best. Ideally, one would run a grid-search on every possible combination of hyperparameters to secure that you have found the global-minimum. However, it is neither efficient nor possible for people without a supercomputer to complete that kind of parameter-search (Chollet & Allaire, 2018). As a result, the application of hyperparameter optimization algorithms has almost become a necessity when building good performing machine learning models.

Essentially, a tuning algorithm can be described as an optimization algorithm trying to minimize a given loss function. This is done by looping over sets of hyperparameters where the goal is to identify the combination that returns the lowest validation error (Bissuel, 2019).

3.2.1 Tuning the Logistic Regression

Compared to the other state of the art machine learning techniques we applied in this thesis, Logistic Regression required less tuning efforts. At the same time, it was crucial for the model outcome that the few parameters that needed to be specified were chosen with respect to the classification problem in hand.

Consistent Parameters

To build a well functioning Logistic Regression classifier, there are two key parameters to specify; the *loss function* and the *solver*.

Loss function:

By giving the *multi_class* parameter the value *multinomial*, we made sure that the Logistic Regression model applied multinomial logistic loss to evaluate its training performance. Essentially, the logistic loss function, or log loss, penalize wrong classifications by calculating the accuracy of the prediction. One can interpret minimizing log loss as maximizing class accuracy, but instead of just generating a single predicted label, it calculates separate probabilities for each distinct class. This makes it possible to penalize the model more when it predicts the actual class to be less likely. In other words, we get an increase in log loss as the predicted probability for the actual class diverges from 1 (Collier, 2015). When we have multiple classes, the log loss function can be defined mathematically as in Equation 3.10.

$$-\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} \log p_{ij} \quad (3.10)$$

Here, N represents the number of samples, M the number of possible classes, y_{ij} indicates if class j is the correct classification for sample i by yielding 0 or 1, and p_{ij} equals the probability for sample i to be labeled with class j . Following this logic, a Log Loss close to 0 would indicate that we have a good classifier (Collier, 2015).

Solver:

To identify the optimal solver, we trained our Logistic Regression model strategically numerous times, while choosing a different solver for each training-run. The solver we used to train our final benchmark model was *lbfgs*, or L-BFGS. L-BFGS is short for *Limited-memory Broyden-Fletcher-Goldfarb-Shanno*, and because it requires a limited amount of computer memory to run, this is a very effective and popular solver to use. This algorithm is based on the same principals as the *Newton's method* for gradient descent (Fu, 2016).

Hyperparameters

As mention above, to build our Logistic Regression benchmark, limited tuning efforts were required. However, in order to avoid overfitting, we experimented with applying different levels of regularization. In all essence, regularization tries to reduce model complexity/flexibility by regularizing or shrinking the model's coefficients towards zero (Gupta, 2017). Since we used *lbfgs* as our solver, the only regularizer we could use was *l2 - ridge regularization*. The argument *C* was tuned in order to identify the "optimal" level of regularization. In other words, the level of regularization yielding the lowest training loss.

3.2.2 Tuning the Gradient Boosting Machine

Consistent Parameters

We utilized the "lightgbm" package in python in order to build a well functioning LightGBM model. In the package, there was one key parameter we needed to specify that would stay consistent throughout the tuning process; objective, number of classes, and loss function.

Objective and Number of classes:

In our data, the 31 wells from the Johan Sverdrup field included 38 distinct rock- formations. We specified the parameters objective and *n_classesasmulti - classificationand38accordingly*.

Loss Function:

To reflect our multi-classification problem, the loss function we used to train our model was *Multi Logistic Loss*. This was the same loss function as for our Logistic Regression Benchmark.

Hyperparameter Optimization

Both our LightGBM models were tuned using a Bayesian optimization algorithm.

Bayesian Optimization:

The Bayesian optimization algorithm was chosen because of its ability to pick the most

promising hyperparameters by considering the results of historical tuning runs. This makes it more effective compared to other tuning methods such as grid- and random search. These methods evaluate different sets of hyperparameters completely uninformed of previous results, forcing them to spend a considerable amount of time evaluating poor hyperparameter combinations (Koehrsen, 2018).

The Bayesian optimization algorithm consists of two main components, an objective function and a probabilistic surrogate model built of the objective function. The algorithm takes individual sets of hyperparameters, feed them to the surrogate model, and picks the set of parameters with the best performance-score. These hyperparameter-values are then applied to the objective function. This process will run in a loop, where the surrogate model will be updated with the actual performance from the objective function continuously throughout the process. In other words, the Bayesian model will then make increasingly informed "bets" on which set of hyperparameters that are likely to perform better on the objective function until the number of specified iterations is reached (Kapil, 2019).

The hyperparameters we have tuned for both LightGBM models are:

1. Learning rate (`learning_rate`)
2. Number of estimators (`n_estimators`)
3. Minimum child weight (`min_child_weight`)
4. Max tree depth (`max_depth`)
5. Feature fraction (`feature_fraction`)
6. Number of leaves (`num_leaves`)

3.2.3 Tuning the Neural Network

Finding a good tuning strategy is especially important when tuning a neural network, as each training run takes significantly longer to run compared to some of the other models used in this project.

Consistent Parameters

Activation Function

As mentioned in Section 3.1.3, an activation function calculates the weighted sum of the inputs and adds a bias to it in order to determine whether a neuron should be activated or not. Without an activation function, the neural network treats the assignment as a linear regression problem and will thereby not be able to learn complex non-linear relationships from the data. In summary, the main purpose of an activation function is to introduce non-linearity into the output of a neuron (V, 2017). For a multi-label classification problem, the softmax activation is recommended. The softmax activation function highlights the largest values and normalizes the outputs so that they sum to 1, meaning each element can be interpreted as class probabilities (Gómez, 2018).

Loss Function

The loss function used for our neural networks was the same as the loss function used in our previously described models. In a neural network, *Multi Logistic Loss* is called *categorical crossentropy*, but it is essentially the same.

Hyperparameter Optimization

Optimizer and Learning Rate

As with other optimization problems, the goal of machine learning is to find the weights that will minimize the loss function. The loss function can be viewed as a high-dimensional optimization landscape in which we are trying to reach the bottom. The optimizer we use can be viewed as the recipe we use in order to search for the global minima. Instead of updating the weights at random, the optimizer will calculate the loss, and after n-iterations, pick the weights that yielded the lowest loss. Next, the optimizer will update the weights in small steps based on given parameters.

The optimizer takes a defined learning rate as input. The learning rate is a hyperparameter that specifies how much to change the model in response to the estimated error each time the model weights are updated. It is important to find the right balance between a high and a low learning rate, as a low rate may result in a long training process and might

lead to the model being stuck in a local minimum and a too rate may result in the model learning a sub-optimal set of weights too fast or an unstable training process (Brownlee, 2019d).

In our models, we experimented with both different learning rates and two types of optimizers, namely Adam and RMSProp. RMSprop is one of the most popular optimization algorithms used in deep learning, only surpassed by Adam (Busaev, 2018). Adam has, in practice, become the default algorithm due to its effectiveness and good results (Brownlee, 2017b). However, there are still some cases in which the RMSprop algorithm yields better results than Adam, and we, therefore, chose to include both as part of our tuning.

Dropout Rate

Another hyperparameter to be tuned is the dropout rate. This hyperparameter tries to prevent overfitting by randomly setting a fraction rate of input units to 0 at each update during training time and thereby introduce random variability (Srivastava, Hinton, Krizhevsky, Sutskever & Salakhutdinov, 2014). Since the subsequent nodes learn from the previous nodes, the random variability introduced by dropout can possibly interrupt the introduction of noisy patterns (Brownlee, 2019b).

Batch size

The batch size refers to the number of samples the model will go through before updating the weights. When a batch is completed, the model will calculate the loss based on the predictions compared to the actual values and adjust the weights accordingly. Consequently, the batch size defines how often the weights of the model will be updated (Brownlee, 2019a).

Epochs

Epochs refers to the number of times the model will run through the entire data set during training. The hyperparameter should be set in a way that leads to a decrease in the loss with every additional epoch. Setting the number of epoch too high might lead to overfitting whereas, it might be difficult for the model to pick up patterns with a number of epochs that are too low (Brownlee, 2019a). In our thesis, we experimented with a different number of epochs. However, a high number of epochs is very time consuming and computationally expensive.

3.3 Model Evaluation

3.3.1 Bias-Variance Dilemma

When working with machine learning, one of the most important decisions you need to make is how to validate your models. Not only does validation give you a better understanding of the models, but it also represents a very valuable measure of how unbiased and generalized the performance is. There are numerous ways of validating the models, but the basis of all validation techniques is how you decide to split the data. By splitting the data, the main objective is to understand how the model will perform when it is introduced to unseen data, like in a real-world scenario. To make the model reliable, it is therefore crucial that the way you decide to split your data reflects the type of data you are dealing with (Grootendorst, 2019). As an example, if you are dealing with time-series data and use a randomized split, the model outcome will be unreliable as future information will be leaked to the training process of the models.

In machine learning, it is normal to measure a model's performance by a prediction error term. At large, this error term can be decomposed into three prediction-error components (Brownlee, 2019c):

- **Bias:** refers to the oversimplified assumptions made to make the target function easier to learn. A model with high bias will pay less attention to the training data and with that return a high loss for both training and validation data. A state referred to as underfitting in the literature. One could picture a linear algorithm. A linear algorithm is fast and easy to interpret, but it is less flexible.
- **Variance:** refers to how much the predicted values vary each time the model uses new data to train a model with high variance is heavily influenced by the details of the training data and does not generalize well to unseen data. A state referred to as overfitting in the literature.
- **Irreducible error:** represents noise that naturally exists in our data, usually as a consequence of incomplete features or inherent randomness in the data. The only way to reduce this error will be to clean the data.

Why is it a bias-variance "trade-off"?

Algorithms with low variance and high bias have a simpler underlying structure and can be described as being less complex, such as regression. On the contrary, algorithms with high variance and low bias have a more flexible underlying structure and can be described as being more complex, such as decision trees. In other words, the level of bias and variance is heavily dependant on the underlying algorithm applied to make the predictions. Since an algorithm can not have both high complexity and low complexity at the same time, there will always be a "trade-off" between bias and variance.

In Figure 3.5, we can see an illustration of the simplified mathematical equation of total error: $Total\ error = Bias^2 + Variance + Irreducible\ Error$. To reach the total error equilibrium, one would seek to find the perfect balance between bias and variance. However, since the underlying target function is unknown, it would be impossible to calculate the exact bias and variance error (Singh, 2018).

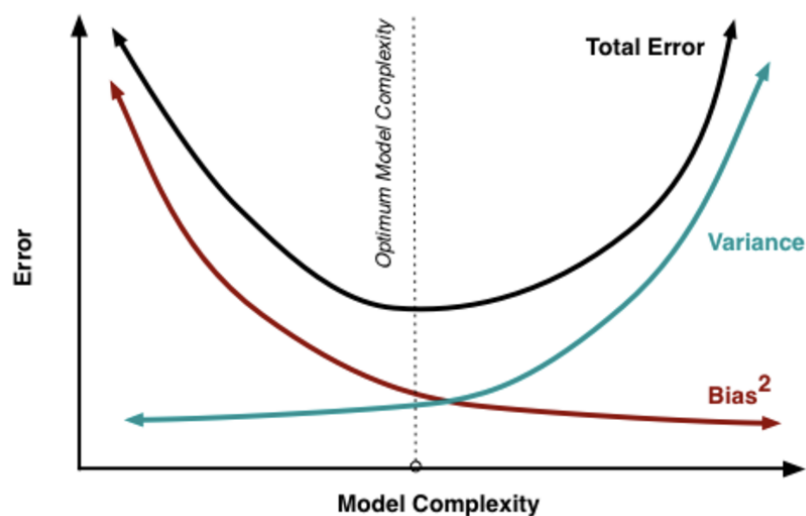


Figure 3.5: Illustration of the relationship between bias and variance. Reprinted from "Bias-variance dilemma?" BY Almaliki (2018).

How is then possible to identify if we have high/low bias or variance?

In practice, we have a common technique to determine if machine learning models are prone to overfit or not. The technique involves comparing the error rate from training data with the error rate from test data. When both training- and test error are high, the model oversimplifies its predictions, meaning it underfits to the training data. When training error is low, and the test error is high, the model has put too much weight on

details in the training data, or in other words, the model is overfitting to the training data (Singh, 2018).

How to deal with over- and underfitting models?

To balance bias and variance in our models, there are some possible solutions, depending on which model you are trying to build. To reduce overfitting, it would be beneficial to collect more data and increasing the size of our training set. This will give the model more data to generalize on and possibly make it easier to deduct patterns that explain the output. However, it is important to note that more training data do not necessarily lead to reduced overfitting. It might also just introduce more noise to the model, which can make it even harder to generalize. Other techniques, such as regularization and feature reduction, can also be helpful to reduce overfitting. By manually removing irrelevant information, you can potentially help the model generalize more efficiently.

To deal with underfitting, the best way would be to build a more complex model construction. This can be adding more trees to a Random Forrest model or to increase the number of nodes and hidden layers for neural networks.

3.3.2 Train/Test Split

A common practice in machine learning is splitting the data set into training-, validation- and test sets. This is crucial, as the only way you can know how well your model will generalize on new data is to actually try it out on unseen data. Through a train/validation/test split, we can train the model on the training set, choose the best performing hyperparameters based on how it performs on the validation set and evaluate the model on data the model has not yet seen. This way, we can get an estimate of the generalization error (Géron, 2019).

In our thesis, we needed to make sure that the three data sets had a realistic distribution of the formations, in order to test the reliability of the machine learning models. As previously discussed, the models are built on the training data, updated continuously based on how well it performs on validation data, and evaluated on the accuracy of test data predictions. Thus, an uneven distribution can lead to models that perform worse on new data, compared to models in which the distribution of the formations was evenly

spread across all data sets. It was also important that there were no formations, or as few formations as possible, that were represented in the test set and not in the train- and validation sets. This is due to the fact that it is impossible for a supervised model to predict a formation it does not know exists. This means that all formations present in the test data should also be present in the validation- and training data sets. The sum of these two steps is normally referred to as a stratified split.

In the Python programming language, this is normally done through a simple function. However, since the goal of the thesis was to predict rock-formations, we needed to consider the fact that these formations have developed over thousands of years. As was discussed in Section 2.1.3, the formations depend on the composition of the materials that go into making it throughout geological history. As a result, formations close to each other are likely to be similar to each other (Brookshire, 2018). This makes the depth-wise sequential information and the location of the well very important when trying to predict rock-formations for each given depth-point of a well. It also means that since the features in each row in a well usually only have a relatively small percentage change from the observation above and below, a normal stratified split would indirectly reveal information about the target variable through the validation process, by spreading observations from the same well across all data sets. The test set, the data set that is supposed to be representing how the data would be obtained and predicted in "real life", would now contain information from wells that are also present in the train- and validation.

If we were to evaluate our models based on a test set that had been stratified in the normal way, we would have obtained artificially good results that would not have been representative of the actual predicting power of the models. When the models then would be used to predict on unseen data, they would no longer have access to information about the correct formation for observations close to the target formation. Hence, it would be much more difficult for the models to make predictions on unseen data. As a result, the accuracy of the models' predictions on a completely blind test set would likely be significantly lower compared to the accuracy of the data stratified in a traditional way. However, this does not mean that it would be impossible for a model, trained and validated on a normal stratified split, to pick up generalized patterns, but it means the accuracy on the test set might be unreliable. This makes it important to hold out a few blind wells,

and if the models perform well on completely blind test data that is representative for the population, it would indicate that the model has generalized well.

To overcome the challenge of splitting the data in so that the formations were correctly distributed, while simultaneously keeping the wells intact, we designed our own solution. This solution was heavily inspired by a [StackOverflow](#)-suggestion (Ali Amin-Nejad, 2019).

Customized Stratified Train/Test Split

The designed solution first calculated the percentage-distribution of the formations in the total data set. This was then used as a target in which the wells were assigned to either the train- validation- or test set, depending on both the percentage-distribution of the formations in the given well and the number of wells already assigned to the respective data sets. The function would make sure that the correct percentage of wells would be distributed across the three data sets, while at the same time keeping an even distribution of formations. This way of splitting the data set led to a train set with 17 unique wells, a validation set with 9 unique wells, and a test set with 5 unique wells. This split, with these exact wells, was consistently used as a starting point for all of our models discussed in the thesis. The python code for the customized solution can be found using the URL-link to the public GitHub repository in Appendix A1.

Lastly, we would like to emphasize the focus we have had on keeping a set of test wells completely separate from our models that were identical for all models. This is a crucial step regardless of which machine learning technique you are applying, but since we have experimented with different degrees ways of splitting the data, it was especially important this thesis. Without testing the models on completely blind wells, we would neither be able to obtain reliable performance measures nor compare our results across models.

3.3.3 Performance Measures

Since some of the formations in data set occurred much more frequently than others, simply computing the accuracy of the predictions could potentially be misleading. When the data set is skewed, a model that only predicts the formation that is most common will

automatically gain an accuracy equal to the percentage distribution of that formation. To resolve this, we used a few different performance measures.

We first calculated the precision and recall of the classifier. The precision is the accuracy of the positive predictions, calculated as True Positive (TP), divided by the sum of True Positive and False Positive (FP). The recall refers to the ratio of positive instances that are correctly detected by the classifier and is calculated as True Positive (TP) divided by the sum of True Positive and False Negative (FN). As can be seen in Equation 3.11, these measures can be combined into a single metric known as the F_1 score, which makes it easy to compare different classification models. As can be seen in the Equation 3.11, the calculation of the F_1 score will give a high weight to low values, meaning the F_1 score will only be high if both precision and recall are high. For multi-class problems, the F_1 score is equal to the accuracy of the predictions (Géron, 2019). This thesis uses the two terms, F_1 score and accuracy, interchangeably.

$$\begin{aligned} \textit{precision} &= \frac{TP}{TP + FP} \\ \textit{recall} &= \frac{TP}{TP + FN} \\ F_1 &= 2 * \frac{(\textit{precision} * \textit{recall})}{(\textit{precision} + \textit{recall})} \end{aligned} \tag{3.11}$$

The same performance measures were also applied at the group-level. As previously discussed, a lot of the formations within a specific group had similar properties, meaning it was relevant to check whether the predicted formation was in the same group as the actual formation. If this was the case, it would mean that the model would be more accurate than if it predicted a formation in a different group. This added extra understanding to the models' predicting power.

3.4 Model Explanation

A lot of times, identifying the reason why a machine learning model makes a specific decision, might be as important as the actual accuracy of the prediction itself. At the same time, the large amount of data used to make accurate predictions requires complex

model structures, making it harder and harder for researchers and experts to interpret the connection between the models' input and output (Lundberg & Lee, 2017). In other words, there is a trade-off between model complexity and interpretability, where researchers either have to choose accuracy or interpretability. With the goal of eliminating this trade-off, several methods have recently emerged in order to make it possible to interpret the connection between the input and the prediction of complex models.

3.4.1 Additive Feature Attribution

Shapley Additive Explanations (SHAP)

For the purpose of our thesis, we have applied the *SHapley Additive exPlanations*, or *SHAP* (SHAP-Developers, nd), framework to better understand how two of our best performing models have made their predictions. *SHAP* is a framework that connects game theory and local explanations to give a consistent measure of feature importance for a given machine learning model (Lundberg, 2017). As illustrated in Figure 3.6, the prediction explanation method *SHAP*, takes some input data together with a complex machine learning model and generates an explanation for a given prediction. *SHAP*, combines several already existing methods, to be able to keep both local and global interpretability of a model. By global interpretability, we are referring to the fact that we are evaluating the feature impact on the complete set of observations in our test data. By local interpretability, we are referring to the fact that we are evaluating feature importance for a specific class.

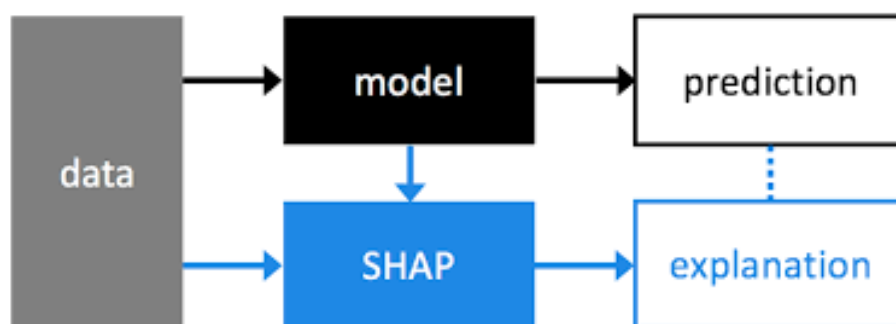


Figure 3.6: Implementation of *SHAP* for model interpretation, reprinted from "Interpretable Machine Learning" (Molnar, 2019).

Even though the *SHAP*-value is based on the old coalition game theory *Shapley Value* introduced by Lloyd Shapley in 1953 (Roth, 1988), it is important to note that the values

represent two different measures. *SHAP* utilizes several components to calculate its values, where one of the main component is *Shapley values*, or *Shapley regression*. The remaining components of the *SHAP* calculations is *Shapley sampling* and quantitative input influence feature attributions, in addition to connections to other attribution methods such as *LIME* and *DeepLIFT* (Lundberg & Lee, 2017). In other words, calculating *SHAP*-values is a very computationally complex process. This section will provide an explanation of *SHAP*, that will allow the reader to understand how *SHAP*-values can be used to interpret model-behavior. However, it excludes detailed explanations of *Shapley regression*, *LIME* and *DeepLIFT*.

The additive feature attribution method *SHAP* can help us explain our model's output, by attributing an effect ϕ_i to all input-features, and summing the total attribution from all features to approximate the output of the original model $f(x)$. Additive feature attribution can be defined as a linear function for binary variables, as defined in Equation 3.12 (Lundberg & Lee, 2017).

$$g(z') = \phi_0 + \sum_{i=1}^M \phi_i z'_i, \quad (3.12)$$

Where $z' \in \{0, 1\}^M$, $M = \text{number of simplified input features}$, and $\phi_i \in \mathbb{R}$. Here, $z'_i = 1$ when the variable is observed, else $z'_i = 0$.

Equation 3.12, shows how the simplified model $g(z')$ can approximate the feature-impact from a complex model $f(x)$ by approximating output z' using the attribution value ϕ_i for each of the individual features. This definition implies that by satisfying the three natural properties, *local accuracy*, *missingness*, and *consistency*, there is only one solution to assign the weights ϕ_i to the linear model $g(z')$ (Lundberg & Lee, 2017).

Natural Property 1 (Local Accuracy)

The natural property *local accuracy* defined in Equation 3.13, says that the output of the simplified model $g(z')$ must equal the output of model $f(x)$ for each prediction being

explained.

$$f(x) = g(z') = \phi_0 + \sum_{i=1}^M \phi_i z'_i \quad (3.13)$$

Natural Property 2 (Missingness)

The natural property *missingness* defined in Equation 3.14, says that a missing feature x'_i from the original model, requires the feature to have no impact in to the simplified model $g(z')$.

$$x'_i = 0 \implies \phi_i = 0 \quad (3.14)$$

Natural Property 3 (Consistency)

The natural property *consistency* defined in Equation 3.15, says that if removing a feature in one model always makes a bigger difference in another model, it should be valued as more important in the first model compared to the second. For two models f and f' , let $f'_x(z' \setminus i)$ represent setting $z'_i = 0$ and the original model $f_x(z') = f(h_x(z'))$

$$f'_x(z') - f'_x(z' \setminus i) \geq f_x(z') - f_x(z' \setminus i) \quad (3.15)$$

This implies that: $\phi_i(f', x) \geq \phi_i(f, x)$

To summarize, *SHAP* provides a unified measure of feature importance where every unique observation of input-data is assigned a *SHAP*-value. Depending on which model you are evaluating, the returned *SHAP*-values will differ. Since the two models, we have analyzed through *SHAP* are built using the tree-based algorithm LightGBM, our predictions are outputted as probabilities. As a result, the *SHAP*-values we will be analyzing is presented in the *log-odds* space. For each class, the sum of these *SHAP*-values ϕ_{ij} (row i and column j) and mean prediction \hat{y} will add up to the total *log-odds* probability P of predicting class x . In other words, \hat{y} can be referred to as the *base value*, which represents the value that would be predicted if we did not know any features for this particular output (Molnar, 2019). This can be defined as seen in Equation 3.16.

$$P_x = \sum \phi_{ij} + \hat{y} \quad (3.16)$$

The *Log-odds* scale is based around 0 and ranges from infinity negative to infinity positive. It represents a binary scenario for each class, where 0 equals 50% likelihood of predicting a given class. In other words, high positive *log-odds* values represent a probability closer to 100% and low negative values a probability closer to 0%.

4 Data Processing

For this project, we were given access to a data-repository containing separate data sets for wells spread across the Norwegian Continental Shelf. Since the models are only able to learn from the input given, the data wrangling and restructuring was one of the most crucial parts of the thesis. The structure of the data sets, the feature engineering, and the choices made when it comes to the processing will affect the models directly. The first section of the chapter refers to the descriptive statistics of the data sets. Proper knowledge of the data set was crucial in order to make the correct decisions in regards to the structure, cleaning, and the feature engineering of the data. The second section discusses the feature engineering while the last section talks about the data restructuring.

4.1 Description of the Data Set

The descriptive statistics that was the basis for the decision made in the thesis are the same as are presented. It is based on the training set only in order to avoid what is known as data snooping bias, overfitting on the test set by, for instance, selecting a particular kind of machine learning model because you have analyzed the full data set (Géron, 2019).

Distribution of Formations

An important discovery was the skewness in the distribution of the formations in the data set. As can be seen in Figure 4.1, most of the formations were either undifferentiated or had no formal name. In fact, those two formations combined accounted for **53.44%** of all the formations in the data set. An uneven distribution is not ideal, as it might lead to the algorithms not being able to predict the less common formations.

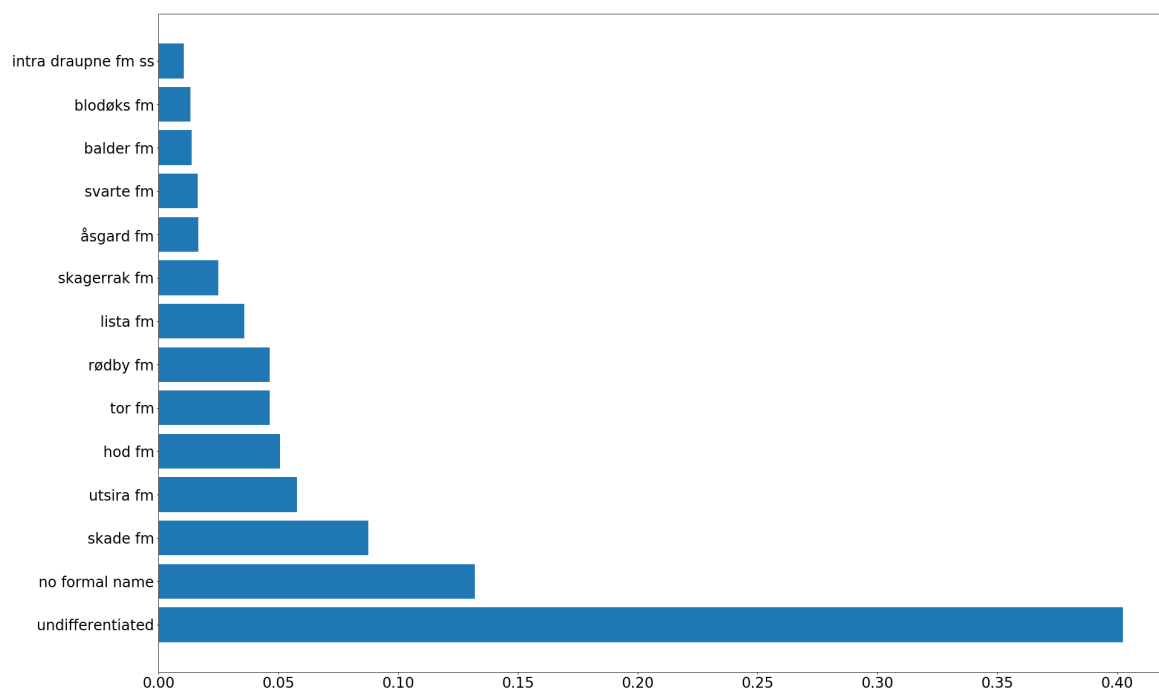


Figure 4.1: Distribution of the formations in the data set. The figure illustrates the skewness in the percentage distribution of the formations. Only the formations that accounted for more than 1% of the total observations are included in the figure. The data set contained 13 additional formations.

The skewness of the formations and the large number of undifferentiated formations were especially a problem since the formations did not necessarily have very distinct feature-characteristics. An example of this can be seen in Figure 4.2. In the figure, three of the input variables are plotted against each other, and the colors represent unique formations in the data set. From the figure, it is clear that the formation from Figure 4.1 that was most common in the data set, the undifferentiated formation, has a lot of variation in the input values. The formation contains values spread across the whole plot. The fact that the variation is so large and that it is the undifferentiated formation that accounts for the highest variation would make it more difficult for the models to separate the formations based on the input.

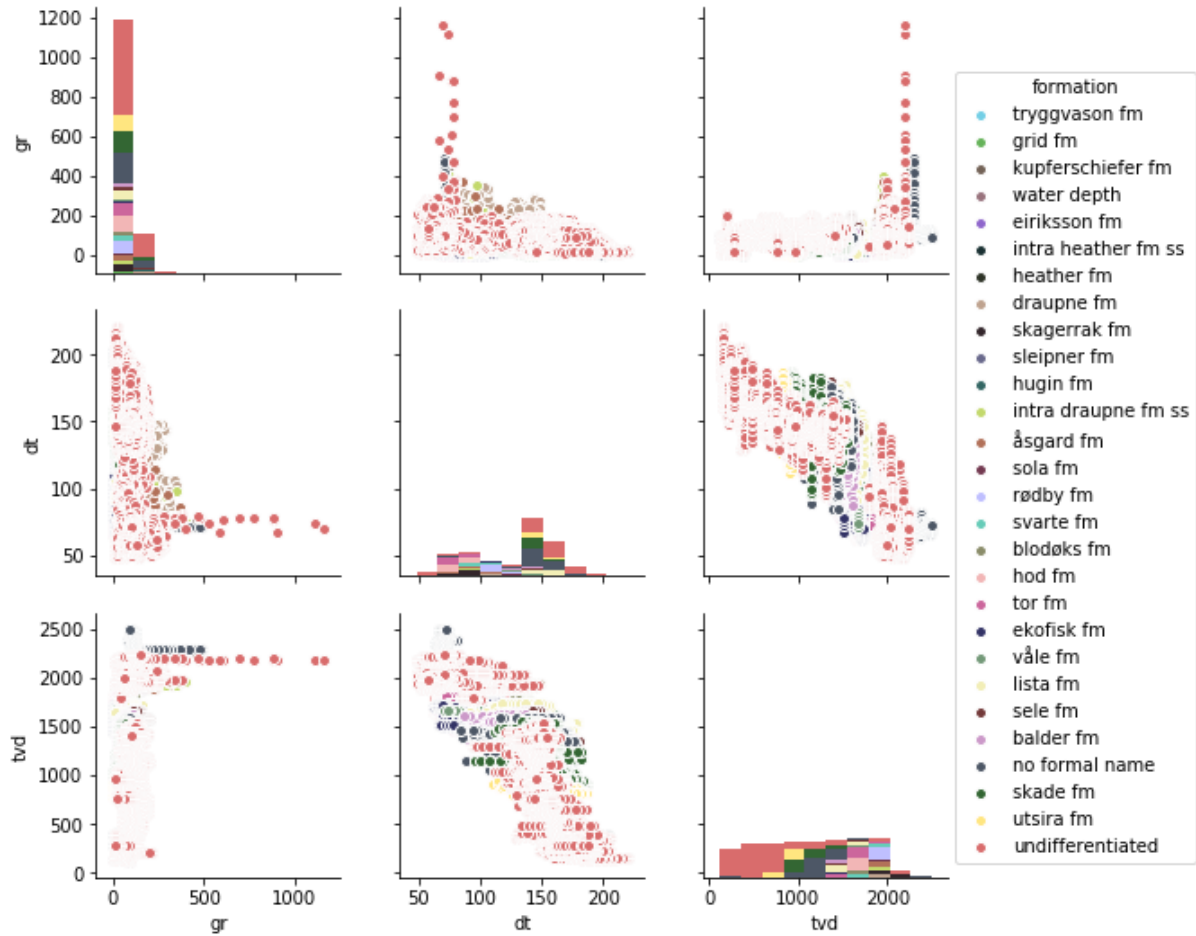


Figure 4.2: Crossplot of original formations. Each color represents a unique formation. The figure shows the input variables gr , dt , and tvd plotted against each other. From the figure it can be seen that the most common formation, *undifferentiated*, has a lot of variation. The formation can be found on the entire specter of values in all the plots.

To resolve this, we labeled all the formations that were either labeled as *undifferentiated* or *no formal name* with the corresponding group for that formation. For instance, an *undifferentiated* formation in the group *nordland gp* would get the label *undifferentiated_nordland gp*. In doing so, the number of formations in the data set went from 28 to 38. We also note that in order to improve the readability of some of the figures, the formations and groups were given a numerical value. A table in which the numerical value and the corresponding formation and group can be found in Appendix A6. The implementation of the additional formations partly solved the problem of the high variability of the *undifferentiated* formations since the groups had more distinct features, compared to the individual formations. This can be seen in Figure A3.2, where the colors of the individual groups are mostly grouped and not spread across the plots. A cross-plot on a group level containing all input variables can be found in Appendix A3. Despite the

solution resulting in lower variability in the *undifferentiated* formation, there were still some issues. The solution did not take into account the possibility that there might be several formations that were undifferentiated within the group. If that was the case, they would all be labeled the same. This would especially be an issue if the formation was labeled correctly in a different well since that would mean two different formations would have two different labels. As there was no way for us to resolve this issue, we implemented the solution above and noted that it might be a cause of error.

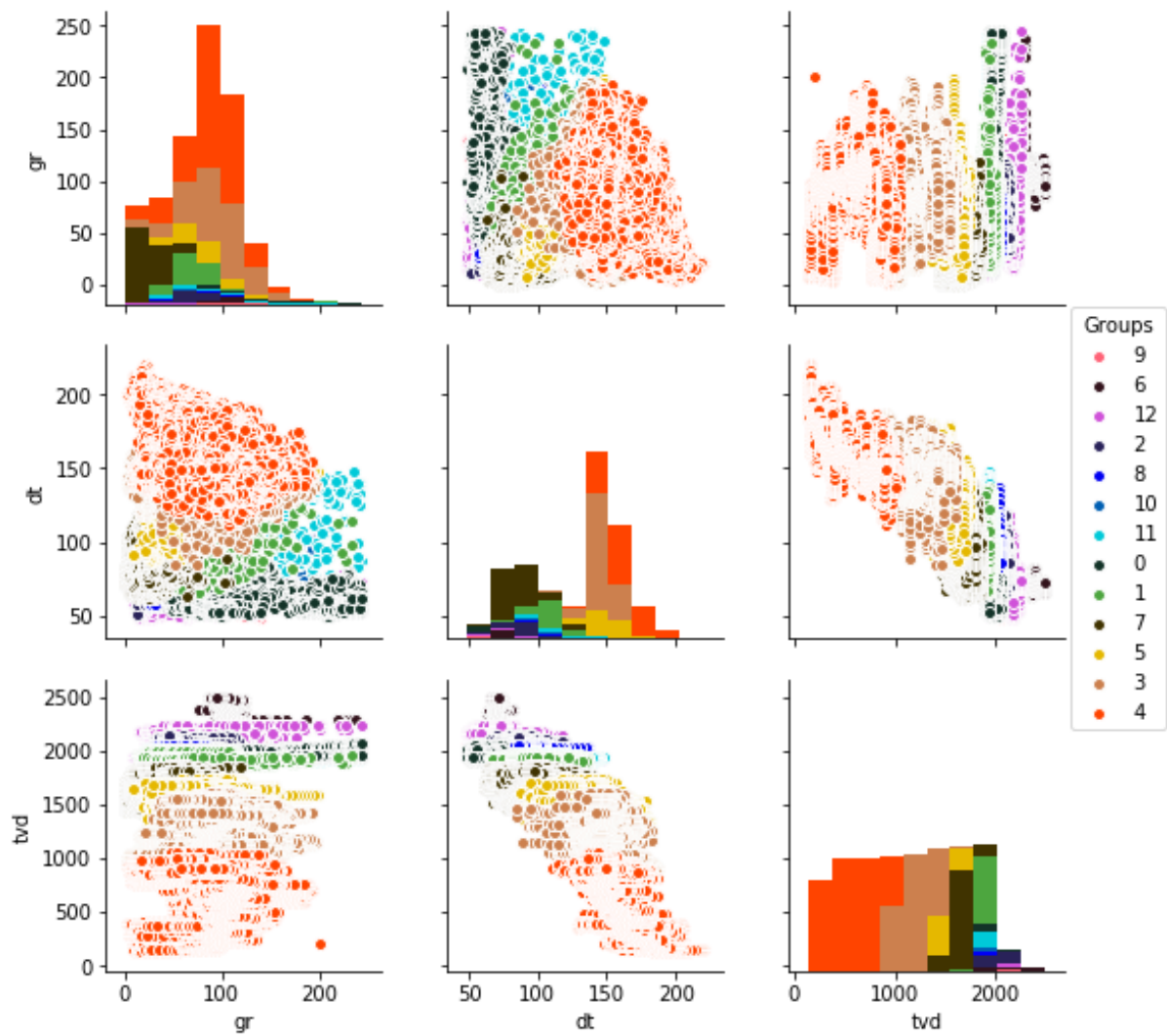


Figure 4.3: Crossplot of the groups in the data set. Each color represents a unique group. The figure shows the input variables gr , dt , and tvd plotted against each other. From the figure it can be seen that the groups have relatively distinct values as the colors are mostly grouped.

Descriptive Statistics

After doing some data exploration we found out that five of the wireline logs; True vertical depth (tvd), Gamma ray log (gr) (Section 2.2.1), Medium Resistivity Log (rmed) (Section 2.2.2), and Deep Resistivity Log (rdep) (Section 2.2.2), were consistently collected for most of the wells. Some of the wireline logs included a large number of missing values, but there were still enough observations that they could be used for some of the models. These logs include Sonic log (dt) (Section 2.2.5), Neutron porosity log (nphi) (Section 2.2.4) and Density log (rhob) (Section 2.2.3). A table containing all variables in the original data set with the corresponding number of missing values can be found in Appendix A2. Through our data exploration, we also observed some obvious errors in the data set. For instance, gr contained some negative values, while rmed contained some extremely large values. As can be seen in Table 4.1, Rmed had a mean of **26.38** and a standard deviation of **1191.49**, but with a median value of **1.3176** it was evident that both the mean and the standard deviation were highly influenced by some of the outliers in the variable. The outliers in the data set can also be seen in the boxplots in Appendix A4.

Table 4.1: Descriptive statistics of the data set before filter was applied

	tvd	gr	rmed	rdep	dt	nphi	rhob
count	342392	342392	337681	337644	204108	141932	141588
mean	1151.6	78.75	26.38	3.44	128.1	0.35	2.25
std	557.23	37.33	1191.49	28.22	33.51	0.15	0.19
min	133.0	-1.24	0.17	0.18	48.12	-0.06	1.37
25%	671.93	55.57	0.89	0.9	95.0	0.22	2.1
50%	1166.66	82.56	1.32	1.32	141.55	0.34	2.25
75%	1632.93	102.96	1.78	1.77	151.92	0.48	2.41
max	2497.9	1163.44	62290.77	2316.71	220.3	1.82	2.99

To solve this problem, we implemented a filter where all values above or below certain thresholds would be removed from the data set. Implementing such a filter can be problematic, as it might lead to correct values being removed, making the models incapable of predicting correct values. On the other hand, outliers and incorrect values might distort the models' ability to generalize well on new data, making it important to find the right balance between a too strict and too mild filter. The values chosen can be seen in Table 4.2 and was set to the 99.95 percentile of the variables, based on recommendations from Pro Well Plan.

Table 4.2: Filter

	Min	Max
gr	0	244.72
rmed	0	1573.66
rdep	0	431.3

As can be seen in Table 4.3, when we applied the filter, the numerical description of the data set changed drastically. For instance, the standard deviation of rmed changed from **1191.49**, before the filter was applied, to **22.52** after the filter was applied, and there were no longer negative values in gr.

Table 4.3: Descriptive statistics of the data set after filter was applied

	tvd	gr	rmed	rdep	dt	nphi	rhob
count	337174	337174	337174	337174	202621	140017	141220
mean	1151.29	78.76	2.69	2.94	128.24	0.35	2.25
std	553.35	36.83	12.53	15.26	33.48	0.15	0.19
min	133.0	0.01	0.17	0.18	48.12	-0.06	1.37
25%	674.94	55.58	0.89	0.9	95.15	0.22	2.1
50%	1165.89	82.8	1.32	1.32	141.64	0.34	2.25
75%	1626.89	103.08	1.78	1.76	151.96	0.48	2.41
max	2494.91	244.7	813.07	431.24	220.3	1.82	2.99

Crossplot matrix

After the filtering of the data set and the inclusion of additional formations, it was important to investigate the results of the data cleaning through a new cross-plot. A cross-plot on formation level, containing all input variables, can be found in Appendix A3 Figure A3.1 indicates that the variability of the formations went down compared to what was evident in Figure 4.2. Other conclusions that can be drawn from the figure is that formation 29 is mostly present in the top part of the wells. Furthermore, since a lot of the colors are overlapping, it is clear that some of the formations contain fairly similar values. This can indicate that the formations might not be linearly separable, an observation that is helpful when determining which algorithms to implement.

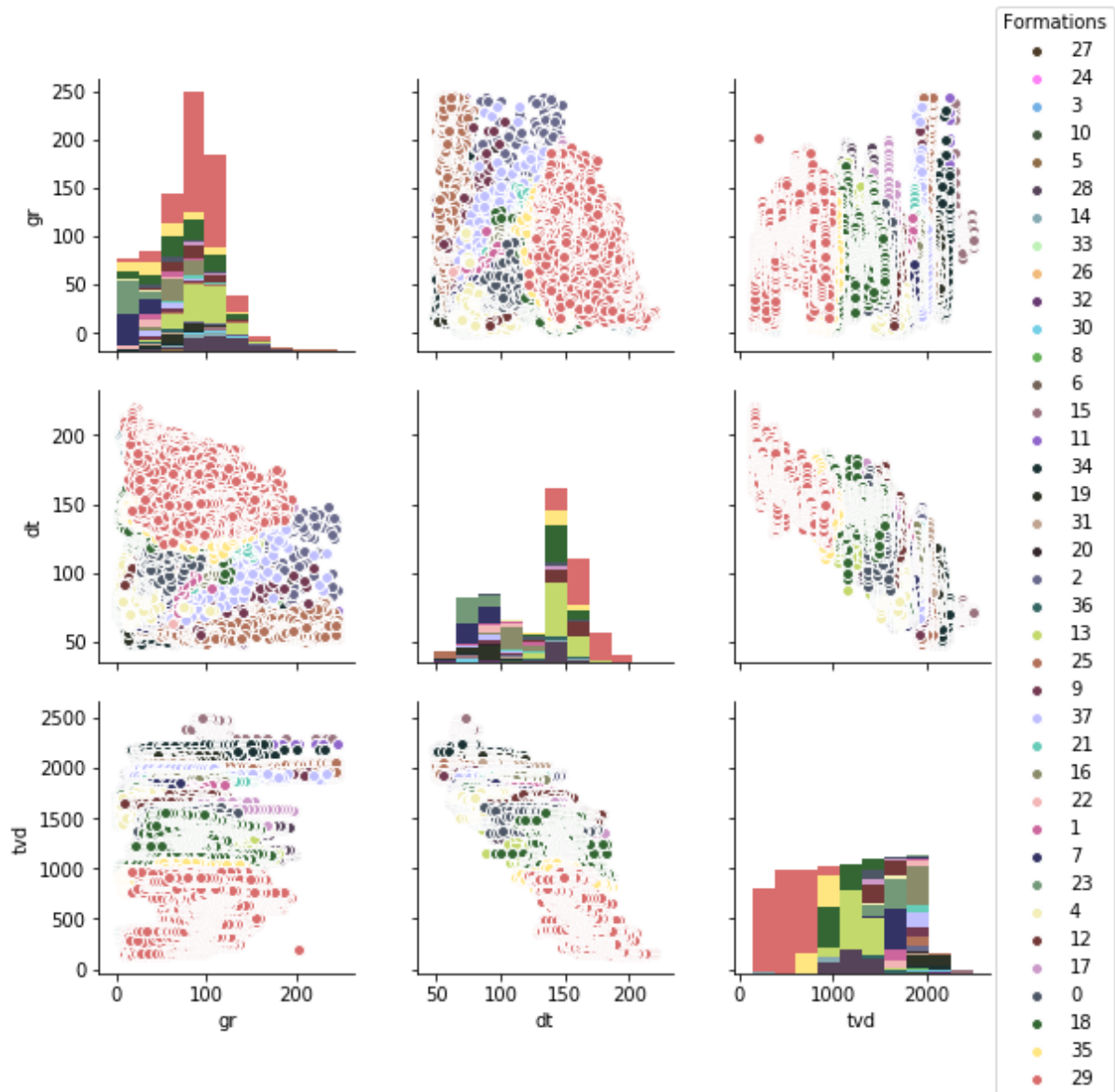


Figure 4.4: Crossplot of all formations. Each color represents a unique formation. The figure shows the input variables gr , dt , and tvd plotted against each other. From the figure it can be seen that the formations are grouped to some degree. For instance, it is evident from the plot in which tvd and gr are plotted against each other that formation 29 is mostly present in the part of the wells where the values of tvd are low, meaning the top part of the wells. However, since a lot of the colors are overlapping it can also be seen that some of the formations contain fairly similar values.

4.2 Feature Engineering

In terms of feature engineering, we applied a few different methods with a goal of improving the predicting power of the models.

Feature Scaling

The fact that some of the variables in the data set had very different scales could potentially cause a problem for some of the machine learning algorithms. Since a lot of the algorithms use Euclidean distance between two data points, the algorithms only take in the magnitude of the feature and neglect the units. This means that the features with high magnitudes will have a stronger influence than the features with low magnitudes (Asaithambi, 2017).

In order to resolve this problem, we used the standard scaling method. As can be seen in Equation 4.1, standardization of a feature is done by subtracting the mean of the feature from all values so that the feature has a mean equal to zero. The values are then divided by the standard deviation of the feature so that the resulting distribution has unit variance. Since standardization is much less affected by outliers, compared to, for instance, min-max scaling, this method was chosen for our data set.

$$z = \frac{x - \mu}{\sigma} \quad (4.1)$$

Feature Transformation

Further, we also transformed the longitude and latitude variables, as described in the Equation 4.2 in order to scale them. Since longitude and latitude represent a three-dimensional space, the highest and lowest value will, in reality, be close to each other. However, this would not be picked up by a machine learning algorithm, meaning that simply using longitude and latitude for machine learning might cause a problem. By mapping longitude and latitude to x, y, and z coordinates, the points that are close in reality will also be close in the data set. When this was done, we could then scale the variables using the standard scaling method described in Equation 4.1.

$$\begin{aligned} x &= \cos(lat) * \cos(lon) \\ y &= \cos(lat) * \sin(lon) \\ z &= \sin(lat) \end{aligned} \quad (4.2)$$

Above and Below

Considering the way rock-formations are formed where formations close to each other are likely to depend on each other, as previously discussed, it could be beneficial for the models to have access to information about the other observations. In order to potentially increase the predicting power, we experimented with the inclusion of some of the features from the observation above and below. For instance, an observation at row 30 would also be fed the gamma ray value for the observation at row 29 and 31.

4.3 Data Restructuring

3-Dimensional Data Set

Since the input of an LSTM network has to be a 3D-array, we needed to restructure the data to fit the specifications seen in Figure 4.5. To do so, we used Lasscock (2019)'s code as inspiration and built on that. When the data was restructured, each batch used information from the four observations above and the four observations below in order to make predictions. Further, the data was structured in a way that takes the depth into consideration. This was motivated by the combination of the fact that the gates in the algorithm allow LSTM-networks to have a memory and the importance of depth when labeling a formation.

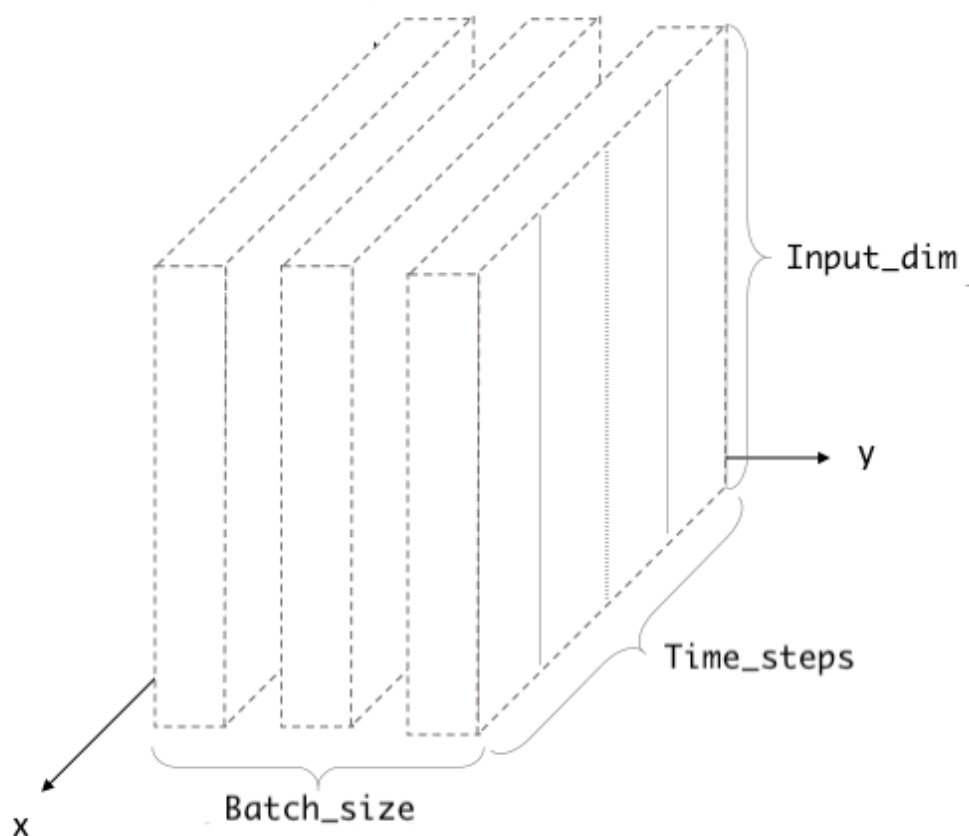


Figure 4.5: Illustration of the input shape for LSTM networks. Reprinted from "Understanding Input and Output shapes in LSTM | Keras" by Verma (2019).

Generator Function

Another issue we encountered when running the LSTM-models, however, was that we quickly ran out of memory and were forced to stop the tuning process. The problem was that each time the model ran through an epoch, one complete pass through the whole training set, the model would not remove the already loaded data, but rather add the data set of the next epoch. As a result of this, fine-tuning the models would have been impossible as it requires running way more epochs than we were able to. To resolve this issue, we created a generator function that would load one batch at a time and remove the batch again. This generator function utilized the same restructuring as described above, but it would do it on each batch instead of the whole data set. As a result, we were able to resolve the memory issue and better tune our LSTM-networks.

5 Analysis and Results

We have built, trained, and evaluated four different machine learning models in order to predict rock-formations as accurately as possible. One Logistic Regression model, which will function as our benchmark, two LightGBM models, and one LSTM model. Each model was trained separately with varying tuning efforts, using different sets of features as input. The following section will describe how each individual model was structured, trained, and we will present the results obtained on the test wells. An explanation of the theoretical framework of the machine learning techniques Logistic Regression, Light Gradient Boosting Machine (LightGBM) and Recurrent Neural Network (LSTM) can be found in Section 3.1.1, 3.1.2 and 3.1.3.

5.1 Experimental Setting 1 - Logistic Regression

5.1.1 Experimental Framework

For our first experiment, our main objective was to build a basic model that could function as a baseline for the rest of our analysis. This benchmark will be used as a reference to evaluate if additional effort, through feature engineering and construction of more complex and computationally expensive models, will be worthwhile with respect to both resources and increased performance. The model was trained using the framework in Table 5.1.

Machine Learning Algorithm:

Logistic Regression

Input Features:

'gr', 'rdep', 'rmed', 'tvd'

Feature Engineering:

'x', 'y', 'z'

Total Data Shape:

(406 666, 20)

Hyperparameter Tuning:

Manual trial and error based on intuition

Objective:

Multi-class

Loss:

Multi logarithmic loss

Regularization:

l2 - Ridge regression

Table 5.1: Experimental setting 1 - Framework

To summarize Table 5.1; The baseline model was created using *Logistic Regression*, an algorithm that is significantly simpler compared to other more recent machine learning techniques. Nonetheless, it is still a powerful algorithm that can help us gain valuable insight with regards to the relationship between our input features and target variable (Shulga, 2018).

The baseline model was given a pre-cleaned and pre-processed data set, following the steps described in Section 4. This left us with an initial set of features consisting of; three distinct well log measures, coordinates of the well, two separate depth measures, and our target variable formation. This will hereafter be referred to our *initial data set*. None of the feature engineered variables were introduced to our benchmark model.

As for all our models, we have used our own stratified split function described in Section 3.3.2 to divide our data set into three parts; training-, validation- and test data. This is done in order to effectively tune and monitor our benchmark model's ability to generalize.

The model was built and trained using the python-package *sklearn* and the function *LogisticRegression* (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg & others, 2011). To tune the model, we used a manual approach based on the intuition described in Section 3.2.1, *Tuning the Logistic Regression*.

5.1.2 Results

In terms of results, the baseline model yielded surprisingly accurate predictions on the test set with a total accuracy of **70.63%**. As can be seen in Table 5.2 The highest accuracy was achieved on well *16/2-7 A* with an accuracy of **85.92%**, whereas the lowest accuracy was achieved on well *16/2-14 T2* with a score of **58.46%**. Table 5.2 also shows that for all the wells, the model had better accuracy on group level. This suggesting that the wrong predictions were usually not too far off since it often predicted formations within the correct group. The total accuracy on group level was **93.47%**, and the accuracy was above **90%** for all wells in the test set. It was interesting to see that the highest accuracy on group level was achieved on the well that had the lowest score on formation with an accuracy of **95.6%**. A possible explanation of this is the previously discussed similarities between formations within the group. It also suggests that despite the relatively poor performance when predicting the formations, the model might have picked up on some trends in the data set.

Table 5.2: Logistic Regression accuracy on blind wells

	LogReg	
	Formation	Group
16/3-7	0.692	0.9312
16/2-14 T2	0.5846	0.956
16/2-11 A	0.6317	0.9384
16/3-6	0.7656	0.9062
16/2-7 A	0.8592	0.9415
Total	0.7063	0.9347

This can also be seen in Figure 5.1, which illustrates both the predicted formation vs. the actual formation and the group that corresponds to the predicted formation vs. the actual group. In Figure 5.1, the well in which the model had the lowest- and the well in which the model had the highest accuracy are presented. Each color in the formation plot corresponds to a unique formation in the data set. This is also true for the groups. A comparison of the predicted values for all models on each well can be found in Appendix A5.

Figure 5.1 serves as a good illustration of the fact that the model performed better on group level than on the formations since the predicted formations typically corresponded

to a formation within the correct group for both wells. When looking at the comparison of well *16/2-14 T2*, we can see that the model, in reality, performed significantly worse on the prediction of the formations than the accuracy of the well suggested. The accuracy was boosted by the fact that approximately the first third of the well consists of the same formation, and this was predicted correctly. However, as soon as the formations changed, the model struggled to predict correctly. It can also be seen that the model had difficulties with the transitions between the formations. This, in combination with the fact that the formation was within the correct group, suggests that for this particular well, the model might not have been able to separate the different formations from each other due to the similar characteristics of the formations in that group. When looking at the comparison of the well in which the model had the highest accuracy, we can see some of the same trends in which the transitions were the most difficult parts to predict. However, compared to the predictions of well *16/2-14 T2*, the transitions are predicted significantly better. The figure suggests that the bottom area of the wells is the most difficult part to predict. This makes sense, considering the fact that the bottom part of the wells usually is more fragmented, compared to the top part of the wells.

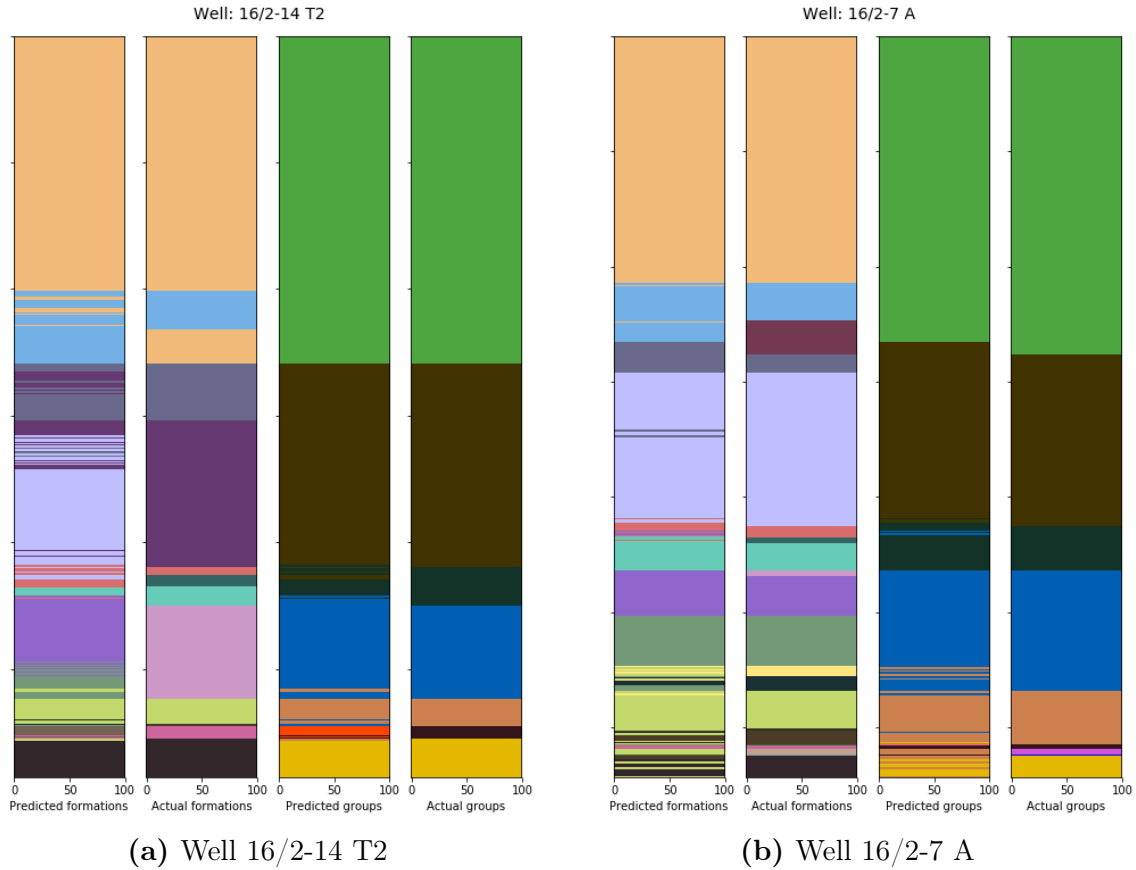


Figure 5.1: The Logistic Regression model’s predicted formations and groups compared to the actual for the best and worse well. Each color represents a unique formation and group. The figure illustrates the areas in which the model predicted correctly and where the model was not able to make the right predictions.

5.2 Experimental Setting 2 - LightGBM 1

5.2.1 Experimental Framework

For our second experiment, we wanted to see if a boosted classification tree would be able to outperform our benchmark model. Here, our main objective was to build a model that was trained on a data structure that replicated how it would be implemented in reality. Our hypothesis was that by replicating a real-world data structure, the model would be able to generalize better, and with that, be able to label our test set more accurately. The framework we used to train the model can be seen in Table 5.3.

<p>Machine Learning Algorithm: LightGBM</p> <p>Input Features: 'tvd', 'gr', 'rdep', 'rmed', 'dt', 'nphi', 'rhob'</p> <p>Feature Engineering: 'x', 'y', 'z', 'gr_above', 'gr_below', 'rmed_above', 'rmed_below', 'rdep_above', 'rdep_below', 'gr_rmed', 'gr_rdep'</p> <p>Total Data Shape: (406 666, 20)</p> <p>Hyperparameter Tuning: Bayesian optimization</p> <p>Objective: Multi-class</p> <p>Loss: Multi logarithmic loss</p> <p>Learning Rate: 0.05</p> <p>Regularization/Dropout: L1</p>

Table 5.3: Experimental setting 2 - Framework

To summarize Table 5.3; The model was built using the tree-based boosting algorithm LightGBM. LightGBM was chosen because it is a high-speed algorithm that requires less memory to run. It also provides a detailed description of feature importance which, gives valuable insight for further analysis.

For this model as well, the data was divided into training-, validation- and test sets using our own splitting function, described in Section 3.3.2.

The model was trained and validated numerous times in a strategic manner. We started off by feeding the model our pre-processed initial data set, as described in Section 4, before we added each of the feature engineered variables, one by one, to see if they added any value to our models. Feature importance was evaluated continuously throughout the training process. This helped us make individual assessments on whether we should keep each of the individual variables as input or not.

Following the tuning process described in Section 3.2.2, a hand-picked set of hyperparameters was tuned using a Bayesian optimization approach, to help us identify the "best" tree structure. After numerous tuning runs on different sub-sets of our complete portfolio of features, the best performing model was trained on 19 input variables; our

complete initial data set, above and below values for gamma ray, medium resistivity, deep resistivity, and re-calculated coordinate values.

5.2.2 Results

With respect to the results, the LightGBM model without a randomized shuffled training and validation split yielded a total accuracy of **77.58%**, a **6.95pp** increase in accuracy compared to our previously discussed benchmark model. As can be seen in Table 5.4, the wells in which the model yielded the highest and lowest accuracy were *16/2-7 A* and *16/2-14 T2*, with accuracy of **88.8%** and **58.3%**, respectively. Furthermore, the model performed very well on the group level, with a total accuracy of **94.23%**. This indicates that even if the model missed on a specific formation, it is in general able to pick up patterns sufficiently enough from our training data that it is able to predict a formation with properties close to the target formation. This is also the case for the worse performing well, well *16/2-14 T2*, where the model had an accuracy of **91.53%** on group level. For most of the wells in the test set, LightGBM 1 obtained a higher accuracy on both formation- and group level, compared to our Logistic Regression benchmark. This suggests that the LightGBM model was able to identify stronger predictors from the training data compared to the Logistic Regression.

Table 5.4: LightGBM 1 accuracy on blind wells

	LGBM 1	
	Formation	Group
16/3-7	0.7132	0.945
16/2-14 T2	0.583	0.9153
16/2-11 A	0.8569	0.9472
16/3-6	0.8109	0.9332
16/2-7 A	0.888	0.9673
Total	0.7758	0.9423

In Figure 5.2, the wells of which the model achieved the highest- and lowest formation accuracy are plotted against each other. By analyzing Figure 5.2, we can also see that the model is able to predict formations within the same group very accurately for both wells. However, by comparing the model's prediction on a formation level, the difference between the two wells becomes more prominent. For well *16/2-14 T2*, the first long sequence of formations is predicted quite accurately, but as we move further down the well, the model

had trouble differentiating between similar formations within the same group. For well *16/2-7 A*, the model is able to distinguish similar formations much better, predicting the long- to mid-length sequences of the same formations very accurately. In general, if we consider the model’s predictions on all blind wells (as illustrated in Appendix A5), there is a clear trend that the transitions between the formations are the most difficult segments to predict, especially for the bottom, more fragmented, part of the well. Therefore, it is impressive to see how precise the model is able to predict the transition-points between the formations for well *16/2-7 A*.

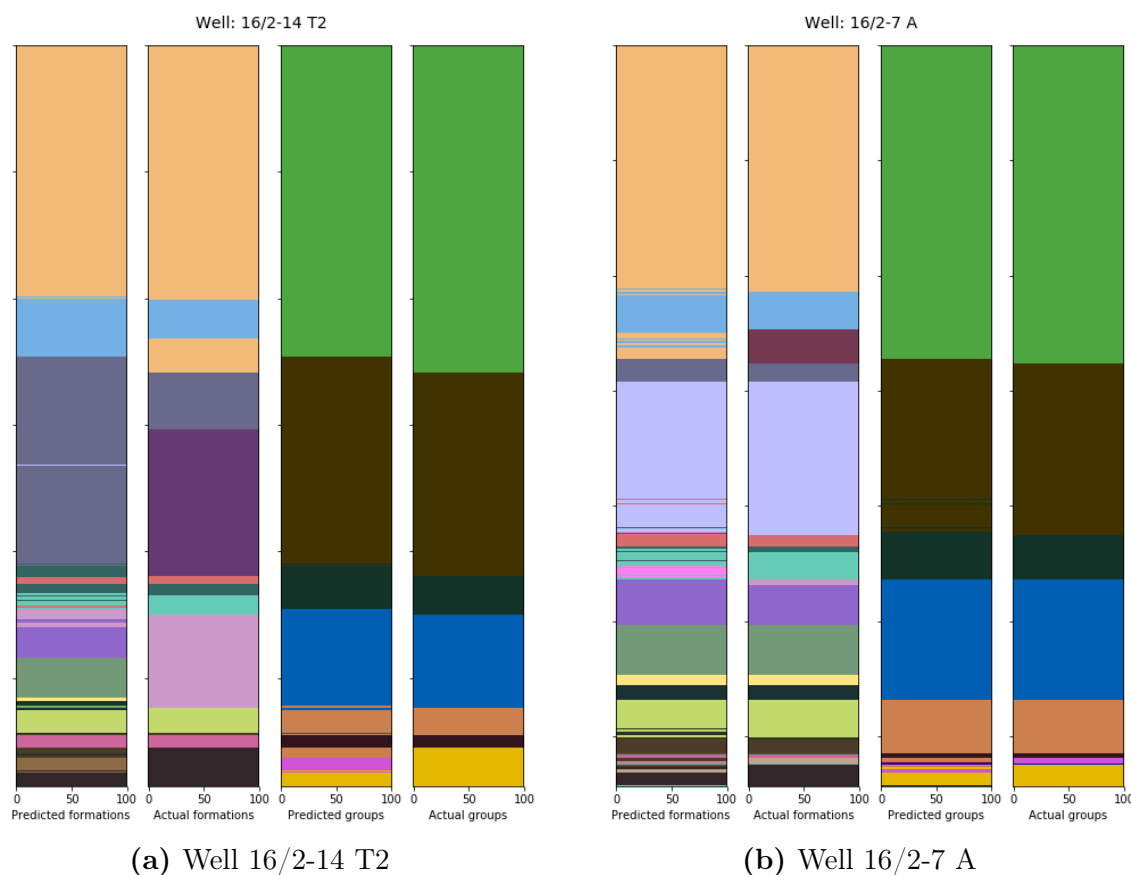


Figure 5.2: LightGBM 1’s predicted formations and groups compared to the actual for the best and worse well. Each color represents a unique formation and group. The figure illustrates the areas in which the model predicted correctly and where the model was not able to make the right predictions.

5.2.3 Model Explanation

Going further, it is also interesting to take a closer look at how the model made its predictions. As discussed in Section 3.4, the package *SHAP* was utilized to analyze how

the different input features influenced the model's predictions. As a quick reminder, *SHAP* helps us make automated calculations of additive feature importance. The outputted *SHAP* value is given at a *log-odds* scale, where the value indicates how much the specific observation added, or decreased, the probability for predicting a particular formation. We will first analyze the predictions on a global level by studying Figure 5.3, before we dig deeper, and look at the feature effects for two formations with high sample frequency in test data by studying Figure 5.4 and Figure 5.5.

To study the input features' effect at a global level, we have plotted the accumulative average *SHAP* values for all input features in Figure 5.3, ordered from highest to lowest influence on the model's predictions. By global level, we are referring to the fact that we are evaluating the model on the complete test set. In this figure, the first 19 formations with the highest cumulative *SHAP*-values were given a unique color while the remaining 19 are marked in turquoise. The formations are colored in descending order, starting with the formation with the highest accumulated mean *SHAP*-values, marked in dark blue.

Figure 5.3 clearly shows that the depth variable *tvd* was the most important feature with respect to the model's ability to separate between the classes. Further, the well logging measures acoustic log (*dt*), density log(*rhob*), neutron porosity (*nphi*), and gamma ray (*gr*), in combination with the two location indicators *x* and *y*, added additional context and predictive power to the model. The specific influence these features' had on the model output will be discussed in greater detail at the end of this section.

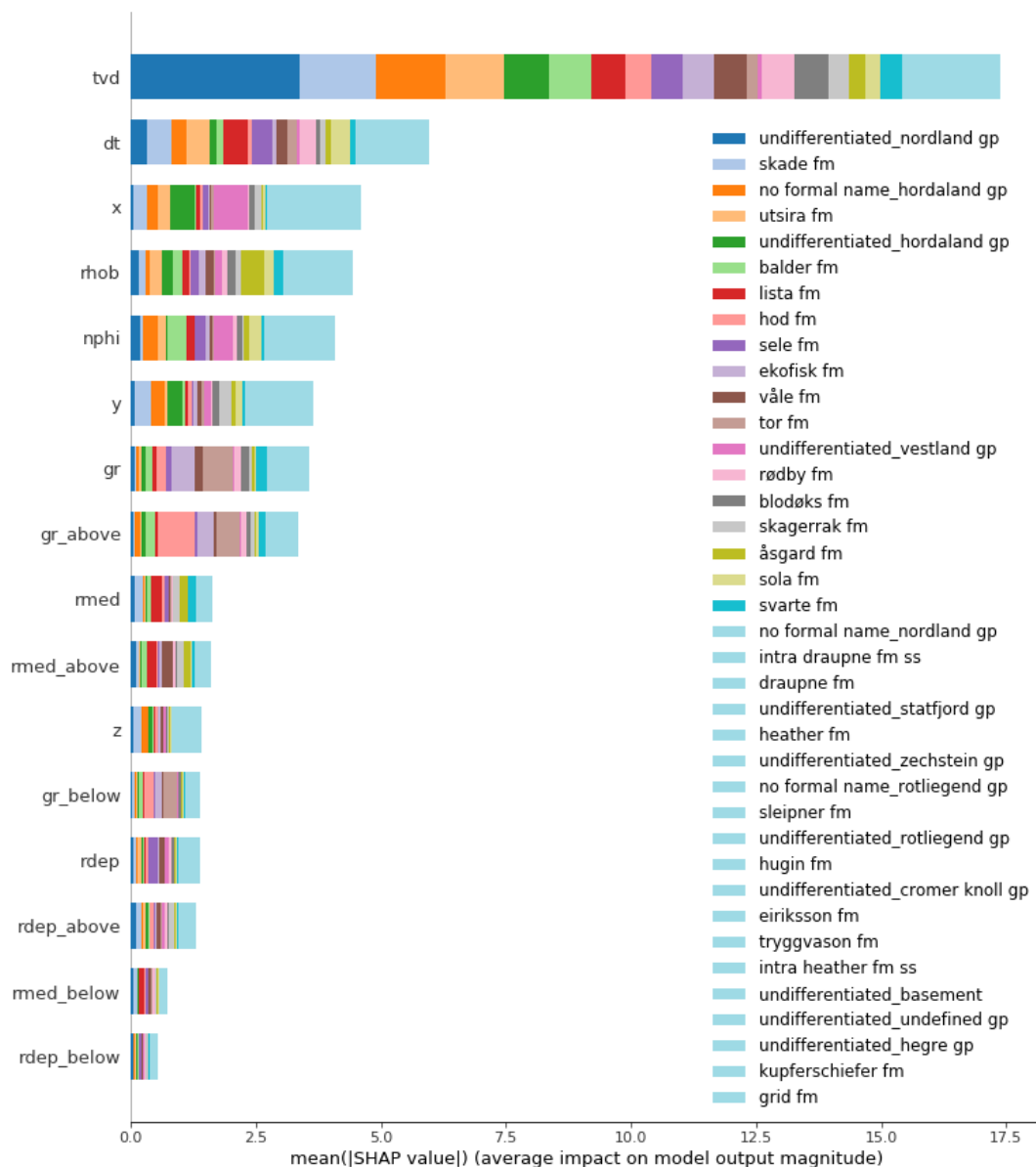


Figure 5.3: The figure illustrates the *SHAP* feature importance for LightGBM 1, measured as mean absolute *SHAP*-values. The first 19 formation is represented by a unique color, the remaining 19 is represented by the turquoise color. True vertical depth was, accumulated for all formations, the most influential feature.

The coloring in Figure 5.3 indicates that, depending on which formation we are talking about, the individual features had a varying impact on the model output. Even though the fluctuations in predictive power for the unique formations were quite low, it is interesting to note that for formation *skade fm*, and especially *undifferentiated nordland group*, the depth measure was particularly important. High depth importance suggests that the two formations generally can be found at a specific depth range of the well. To further investigate such formation-characteristics, we took it a step further and plotted the feature

importance for each variable against two of the formations with the highest sample frequency in our blind test data.

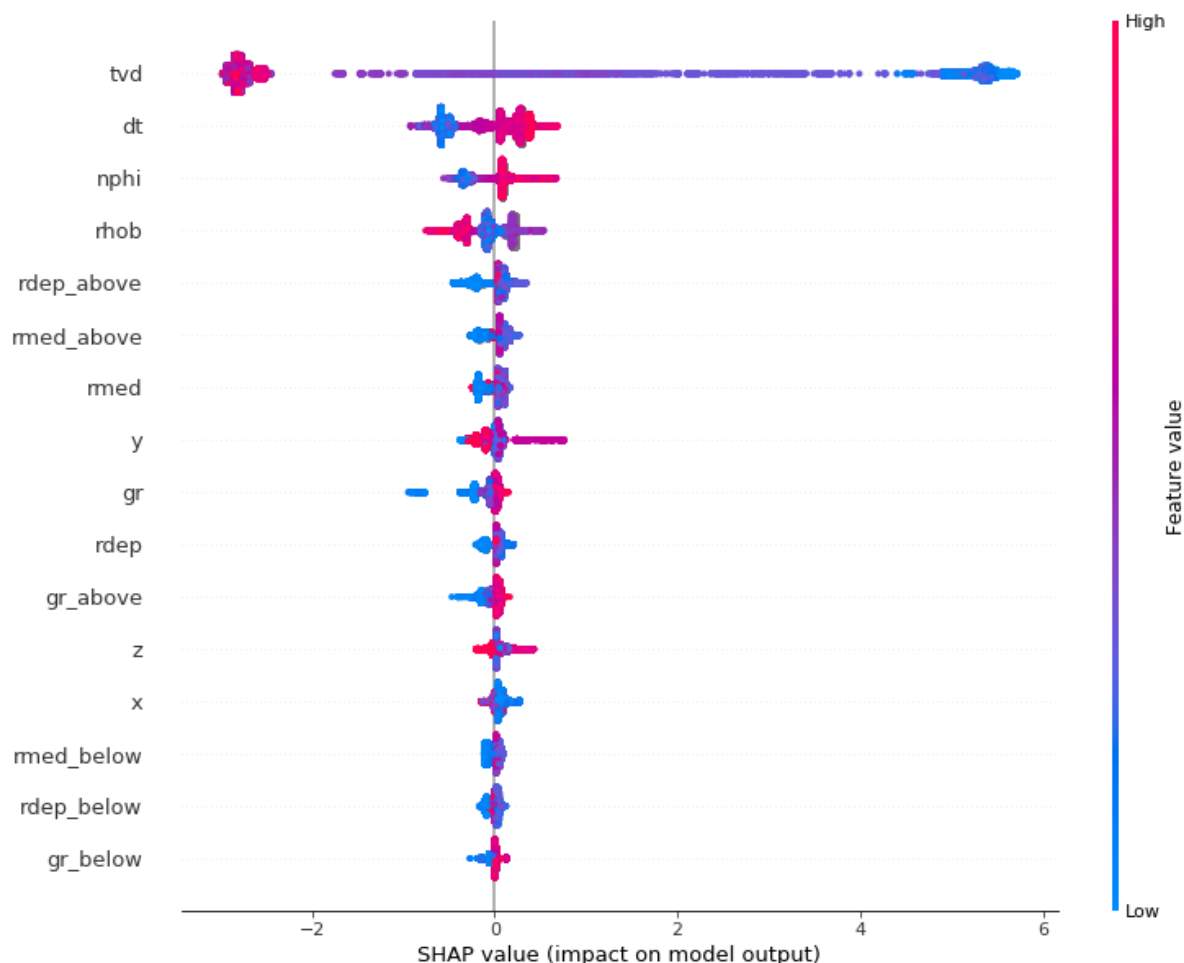


Figure 5.4: The SHAP summary plot for LightGBM 1 illustrates the relationship between the original feature value and the impact on the predicted class *undifferentiated nordland gp*. The feature impact is shown on the horizontal dispersion on the x-axis. The coloring of the points indicates if the original feature value was high (violet) or low (blue). As an example, a low original feature value for true vertical depth increase the models predicted probability of classifying the observation *undifferentiated nordland gp*. On the other hand, a high feature value for true vertical depth, would push the predicted probability for class *undifferentiated nordland gp* towards 0.

Figure 5.4 gives us a detailed look at the model’s feature importance for the formation *undifferentiated nordland gp*. By studying the SHAP values, we are now able to identify at what depth-range of the well *undifferentiated nordland gp* normally can be found. The most influential feature *tvd*, has a clear negative relationship to the formation *undifferentiated nordland gp*, where low depth values increase the probability for the observed value to be classified as *undifferentiated nordland gp*, and vice versa. Next, we can also see that the model identified a positive relationship between the acoustic log *dt*, and neutron porosity

nphi. To summarize, we can say that an observation close to the surface of the well with a high measured value of *dt* and *nphi*, and a mid-range value of *rhob*, would make it very likely for the model to output *undifferentiated nordland gp* as the predicted formation.

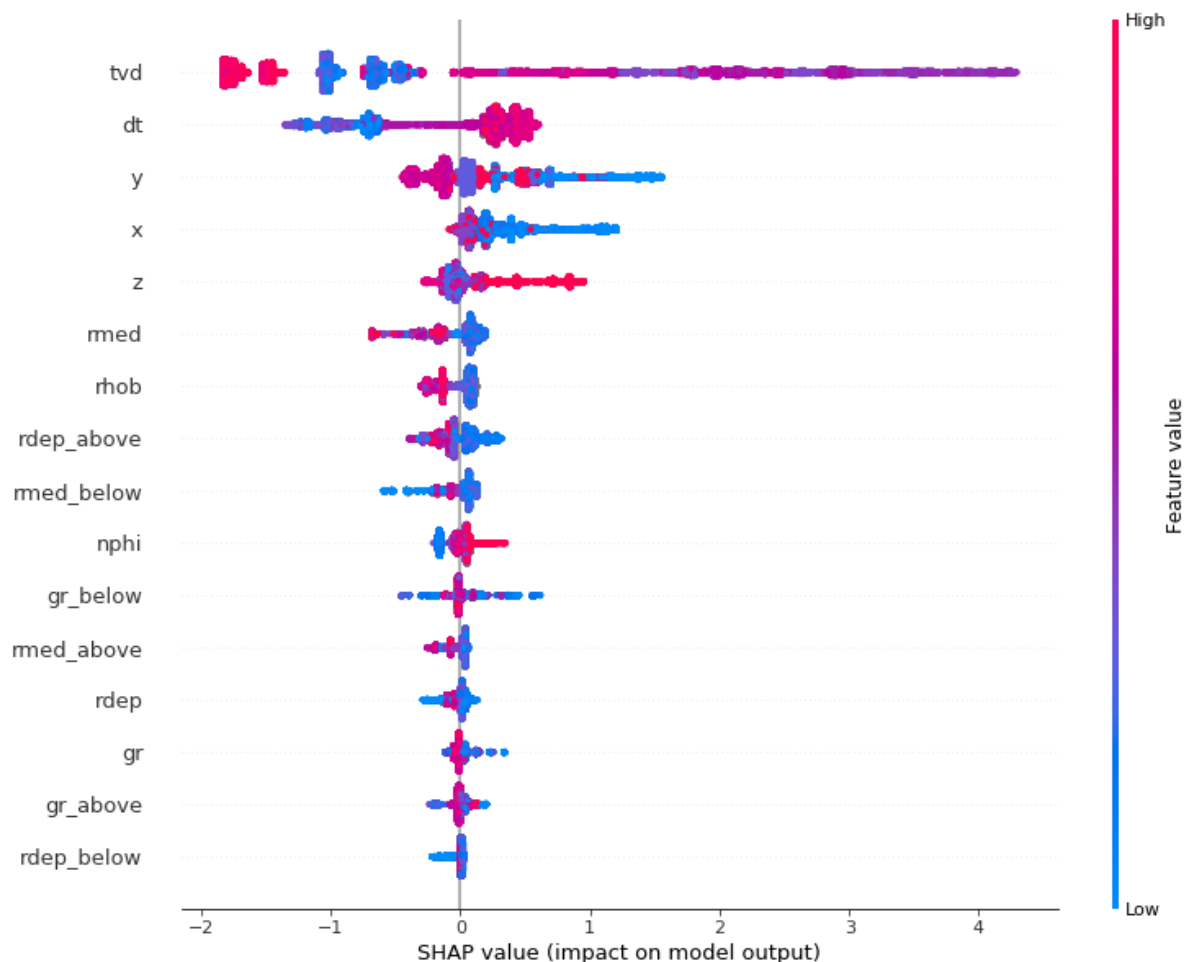


Figure 5.5: The SHAP summary plot for LightGBM 1 indicates that medium true vertical depth values increase the models predicted probability of classifying the observation *skade fm*.

Going further, we can take a brief look at the feature importance for the formation *skade fm* in Figure 5.5. Here, it is interesting to notice that all three location indicators *x*, *y*, and *z* is among the strongest predictors for *skade fm*. In other words, this formation seems to be location-specific. To summarize, certain locations given by *x*, *y*, and *z*, mid-range depth measures, and high values of *dt*, would make it very likely for the model to output *skade fm* as the predicted formation.

5.3 Experimental Setting 3 - LightGBM 2

5.3.1 Experimental Framework

For our third experiment, our main objective was to re-structure our training and validation data to get as many samples of each individual formation as evenly distributed as possible. This would give the model more examples of each formation to train on, which again could help the model pick up more patterns with respect to the relationship between our input features and the individual formation layers. By evenly stratifying our training- and validation data this way, the series of observations for each well was randomly split across the data sets. As discussed in Section 3.3.2, this gave our model bias because it was given information about nearby formations through the training and validation process. To get reliable test results, it was therefore crucial that we kept the original structure of our test data. If we were to stratify our test data in a similar manner, we would most likely get unrealistically accurate results, as the model would have been given information indirectly that it would not have access to in reality. The framework used to train the model can be seen in Table 5.5.

Machine Learning Algorithm:

LightGBM

Input Features:

'tvd', 'gr', 'rdep', 'rmed', 'dt', 'nphi', 'rhob'

Feature Engineering:

'x', 'y', 'z', 'gr_above', 'gr_below', 'rmed_above', 'rmed_below', 'rdep_above', 'rdep_below', 'gr_rmed', 'gr_rdep'

Total Data Shape:

(406 666, 20)

Hyperparameter Tuning:

Bayesian optimization

Objective:

Multi class

Loss:

Multi logarithmic loss

learning rate:

0.05

Regularization/Drop-out:

L1

Table 5.5: Experimental setting 3 - Framework

To summarize Table 5.5; This model was built using the tree-based boosting algorithm LightGBM. We started off with the same training-, validation- and test split as for all the other models, but in order to get an even distribution of our target variable, we applied the function *StratifiedShuffleSplit* from *sklearn* and specified formations as our stratifying criteria (Pedregosa et al., 2011). Our initial test set remained completely untouched throughout this data restructuring process.

Following the tuning process described in Section 3.2.2, the model was trained and validated using a Bayesian optimization approach, while taking the best performing subset of features from Experimental Setting 2 in Section 5.2 as input.

5.3.2 Results

As can be seen in Table 5.6, the LightGBM model with a randomized training and validation split outperforms the previously discussed models with respect to both formation- and group accuracy. The model yielded a total accuracy of **79.17%**, which is an **8.54pp** increase from our benchmark model, and a **1.59pp** increase from our first LightGBM model. The formation-accuracy on individual wells are mostly higher than LightGBM 1, except for the first two wells *16/3-7* and *16/2-14 T2*. For the other wells the increased formation-accuracy was between **1.29** and **5.28** percentage points, compared to LightGBM 1. Similarly to the previously described models, the best performance was returned for well *16/2-7 A*, now with an accuracy of **91.84%**. Going further, it was also interesting to see that even if the formation-accuracy decreased for well *16/3-7* and *16/2-14 T2* compared to the first LightGBM model, we saw an increase in group-accuracy with **1.66pp** and **2.98pp**, respectively. This indicates that even though the classifier on average had more wrong predictions than LightGBM 1, the formations predicted were mostly formations with similar characteristics to our actual formations, as they were in the same group.

Although the accuracy metric provides useful information, we have in the earlier sections of our analysis seen that it, in some cases, can be a bit misleading. As a result, we will now take a closer look at the model's actual performance on the wells with the highest- and lowest accuracy. The results for *16/2-14 T2* and *16/2-7 A* are illustrated in Figure 5.6.

Table 5.6: LightGBM 2 accuracy on blind wells

	LGBM Stratified	
	Formation	Group
16/3-7	0.7091	0.9616
16/2-14 T2	0.5698	0.9451
16/2-11 A	0.8698	0.9651
16/3-6	0.8637	0.9559
16/2-7 A	0.9184	0.9807
Total	0.7917	0.9621

By studying the worst-performing well *16/2-14 T2* in Figure 5.6, we can see that the model seems to be predicting the transition-points between formations more accurately compared to LightGBM 1. On the other hand, it also seems to have slightly more difficulties identifying distinct characteristics for formations within the same group. This means that the model, on average, miss-predicts the target formation to be a similar formation within the same group. This was also discussed, considering our recent analysis of the accuracy in Table 5.6. For well *16/2-7 A*, the LightGBM algorithm continues to perform impressively well. After training the model on a stratified data set, the model was able to classify both formations and transition-points between formations very accurately, even for the lower, more fragmented, part of the well.

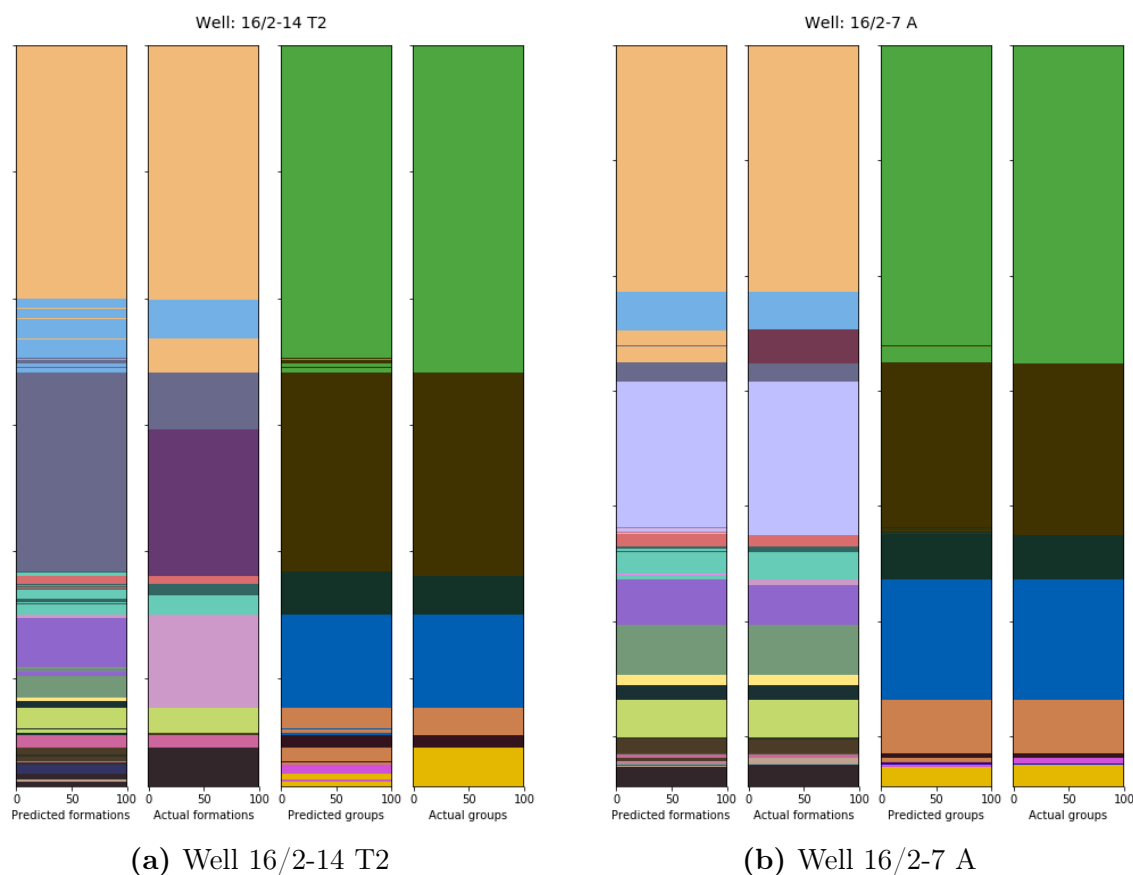


Figure 5.6: LightGBM 2’s predicted formations and groups compared to the actual for the best and worse well. Each color represents a unique formation and group. The figure illustrates the areas in which the model predicted correctly and where the model was not able to make the right predictions.

5.3.3 Model Explanation

To learn more about LightGBM 2’s behavior, we looked at how the individual features influenced the model’s predictions. Similar to the model explanation of LightGBM 1, we started by analyzing the predictions at a global level. Next, we studied the individual feature effects for the formations *undifferentiated nordland gp* and *skade fm*, the same two formations as analyzed before. This way, we could effectively compare the feature importance between the two tree-based models.

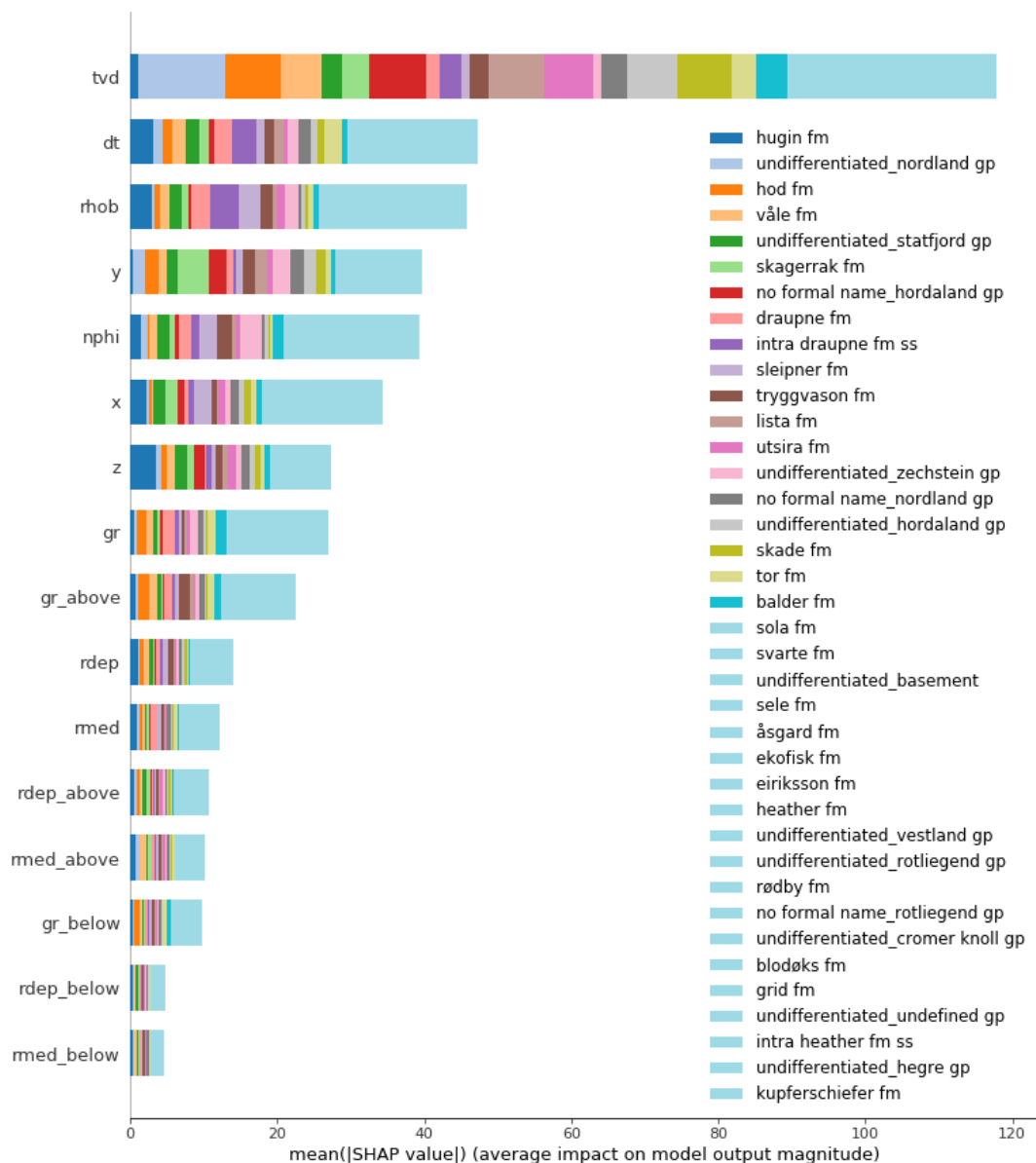


Figure 5.7: The figure illustrates the *SHAP* feature importance for LightGBM 2, measured as mean absolute *SHAP*-values. The first 19 formation is represented by a unique color, the remaining 19 is represented by the turquoise color. True vertical depth was, accumulated for all formations, the most influential feature.

Figure 5.7 shows the input features' effect at a global level. It was interesting to see the significant increase in *SHAP*-values for all features compared to LightGBM 1. This suggests that the individual features now have become stronger predictors, each influencing the model more compared to previous model LightGBM 1. From the figure, it can be seen that for this model as well, the depth feature *tvd* had the most substantial cumulative influence on the model's predictions. However, for the next eight features, there were some minor re-ordering of the most important features. The most noticeable change was that

all the three distance measures were ranked as the fourth, sixth, and seventh strongest predictors for the model. This indicated that the model found more prominent patterns concerning the relationship between location and the unique formations.

The coloring in Figure 5.7 indicated that there were few changes compared to the previous model, LightGBM 1. First and foremost, it is worth noting that the formation *undifferentiated norland gp* (marked in light blue) now was influenced less by the depth feature *tvd* relative to the other features. However, it was still, by far, the most important feature for that formation, meaning for certain depth points, it is likely that the observation is *undifferentiated norland gp*.

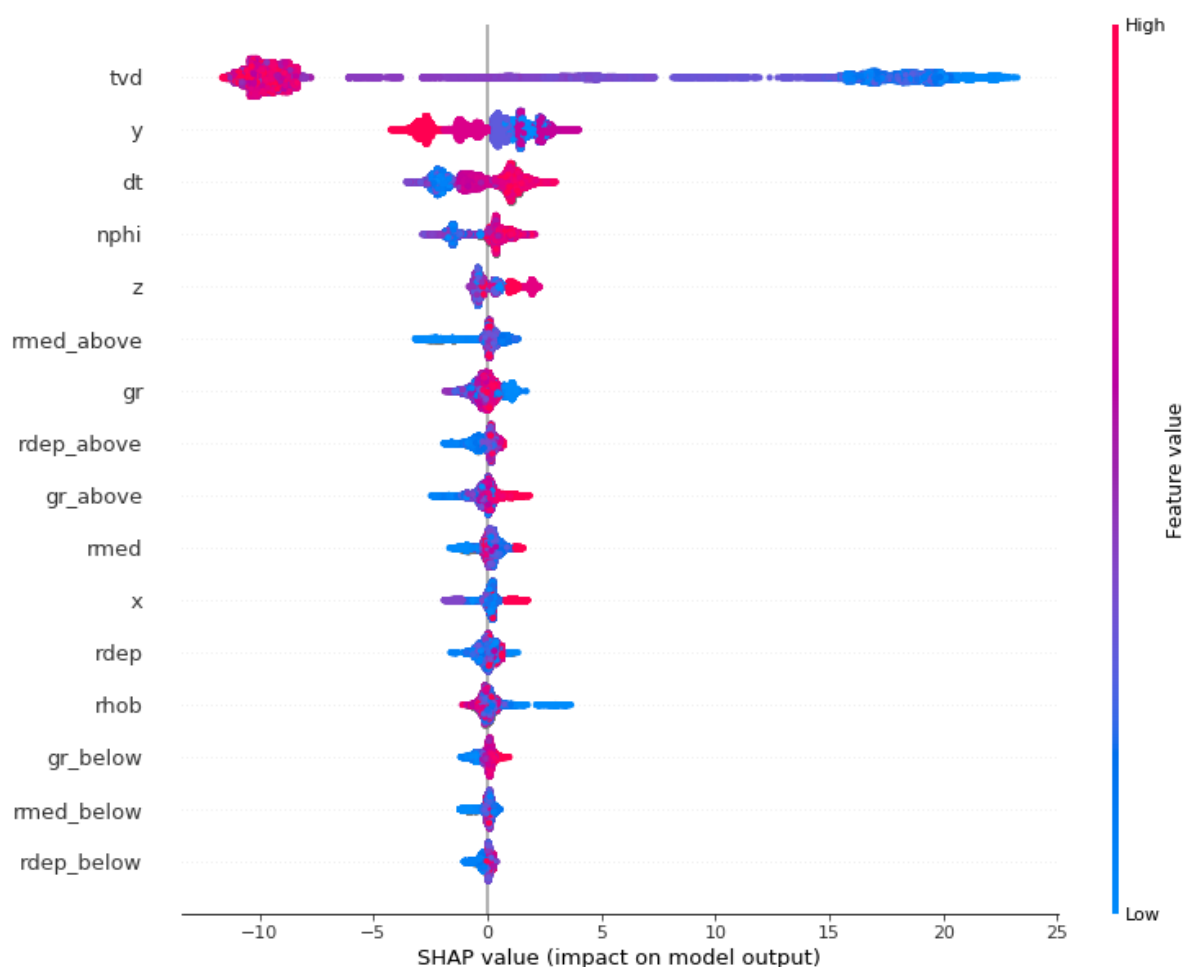


Figure 5.8: The *SHAP* summary plot for LightGBM 2 indicates that low true vertical depth values increase the models predicted probability of classifying the observation *undifferentiated nordland gp*.

To further investigate these formation-characteristics, we plotted all *SHAP*-values against each individual feature. To be able to compare the results, this was plotted for the same two formations as for the model, LightGBM 1. Figure 5.8 depicts the model's

characteristics of formation *undifferentiated nordland gp*. The color scale goes from red to blue, where red represents a high value of the given observation, and blue represents a low value. Considering the most influential feature *tvd*, the model found a clear negative relationship between depth and probability when predicting *undifferentiated nordland gp*. For the remaining features, the most noticeable difference was the fact that the location features *y* and *z*, were of significant importance. To summarize, an observation close to the surface of the well, high measures of *dt* and *nphi*, in addition to a medium to low value of *y* and a high value of *z*, will push the predicted probability for the model to output *undifferentiated nordland gp* towards 100%.

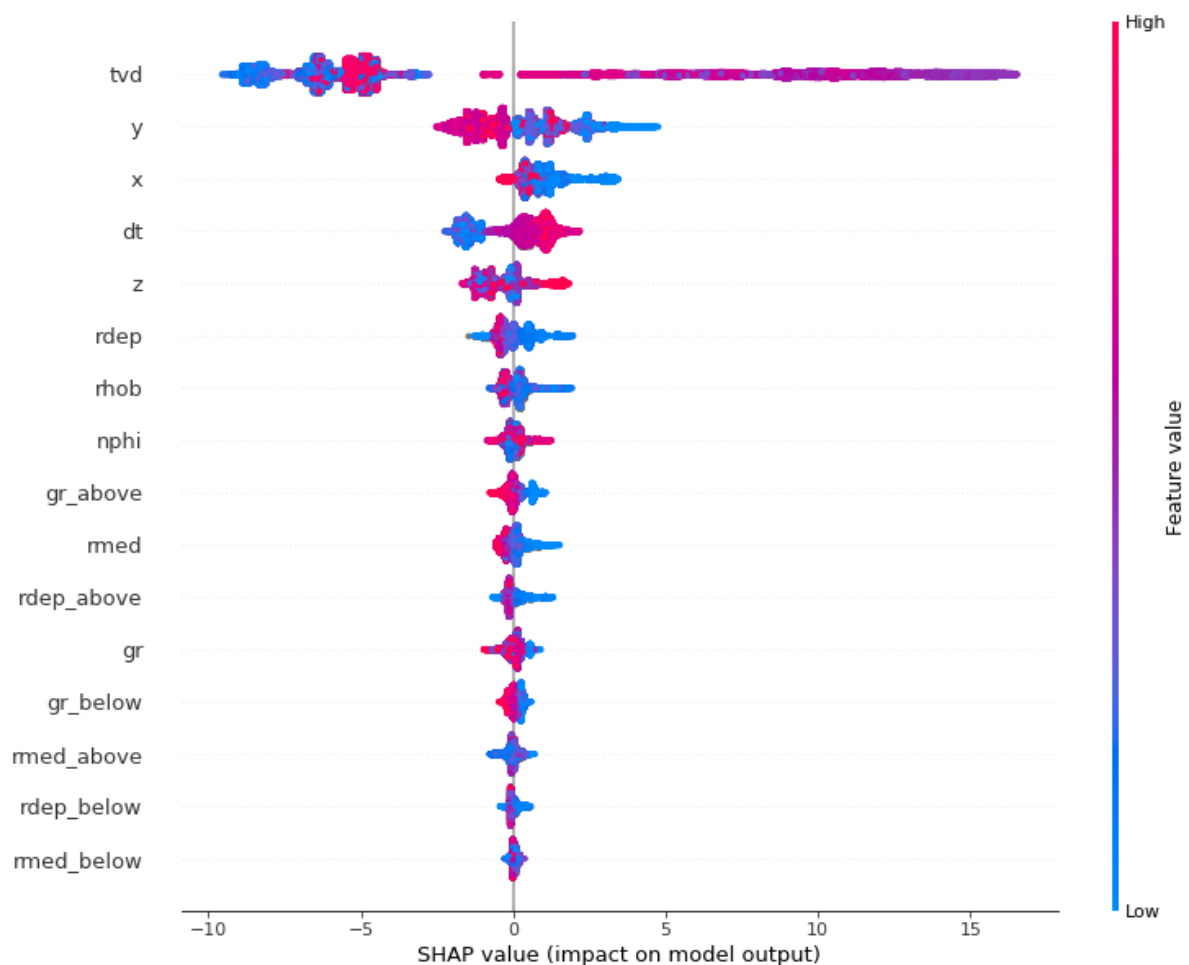


Figure 5.9: The SHAP summary plot for LightGBM 2 indicates that medium true vertical depth values increase the models predicted probability of classifying the observation *skade fm*.

Next, for formation *skade fm* in Figure 5.9 the individual features' relative effect on the predicted probability did not changed much. We can see that location, thorough feature *y* and *x*, have climbed up a few places, giving location an even bigger impact on the model's

predicted probability for the observation to be classified *skade fm*. To summarize, certain locations given by x , y , and z , a medium to high depth value, high dt , in addition to a low $rdep$ and $rhob$, would push the model's predicted probability for classifying *skade fm* towards 1.

5.4 Experimental setting 4 - LSTM

5.4.1 Experimental Framework

For our neural network experiment, we wanted to see if an LSTM-network would be able to pick up on the importance of the data structure in a well. Here too, our main objective was to build a model which that trained on a data structure that replicated a real-world scenario as close as possible. Our hypothesis was that by replicating a real-world data structure, the model would be able to use some of the information geologists are using in regards to how the formations were formed. Since formations close to each other are likely to depend on each other, a model with memory might be able to pick up on the depth-wise sequential information and the importance of the location of the well. The framework we used to train the model can be seen in Table 5.7.

Machine Learning Algorithm:

LSTM

Input Features:

'tvd', 'gr', 'rmed', 'rdep'

Feature Engineering:

'x', 'y', 'z'

Total Data Shape:

(337 280, 8, 9)

Hyperparameter Tuning:

Random grid search

Objective:

Multi-class

Loss:

Categorical Crossentropy

Learning Rate:

0.01

Drop-out:

0.5

Batch Size:

128

Activation Function:

Softmax

Optimizer:

Adam

Network Structure:

Layer1: LSTM

Layer2: Dropout

Layer3: Dense

Table 5.7: Experimental Setting 4 - Framework

To summarize Table 5.7; The model was built using an LSTM-algorithm. LSTM was chosen due to the promising results it has shown on time series data and the fact that it could potentially pick up on patterns in the data set as a result of the memory gates discussed in Section 3.1.3.

In terms of features, the original features were chosen, where longitude and latitude were represented as x, y, and z, as discussed in Section 4.2. The train- and validation set had a shape of (337 280, 8, 9), where 337 280 refers to the number of observations, 8 to the number of variables used to predict, and 9 represents the observations each row in an observation has access to, in addition to the row itself. For instance, when predicting the formation in row 10, the model has access to the information in the variables in row 10 plus the information in the variables in row 6-9 and 11-14.

The network consisted of an LSTM-layer, followed by a dropout layer with a rate equal to 0.1. The output layer of the model was a dense layer with a *softmax* activation function. The optimizer chosen for the network was Adam.

5.4.2 Results

As can be seen in Table 5.8, the LSTM model performed significantly worse than all the previously discussed models. The model had the lowest accuracy of all models, with a total accuracy of **62.72%**. This was **16.45pp** lower than the highest performing model, and compared to the benchmark model, which obtained an accuracy of **70.63%**, it was **7.91pp** lower. Table 5.8 shows that the lowest accuracy on the individual wells in the test set came from the predictions of well *16/2-14 T2*, and the highest accuracy came from the predictions of well *16/3-6* with accuracy of **42.06%** and **77.76%**, respectively. The LSTM model's accuracy on these wells was significantly lower than the accuracy of the other models obtained on the same wells. It is also interesting to see that despite the fact that the accuracy on the group level was higher than the accuracy on the formations, even these are far below the ones obtained by the other models we explored. The total accuracy on group level was **79.84%**, an accuracy that was **13.63pp** lower than the total accuracy of the benchmark model and **16.37pp** lower than the highest obtained group total accuracy. Furthermore, whereas all the accuracy, obtained by the benchmark model on individual wells, were above **90%**, the LSTM model did not have a single accuracy above this score. The highest overall score came from the group level on well *16/3-6* and was **85.06%**.

Table 5.8: LSTM accuracy on blind wells

	LSTM	
	Formation	Group
16/3-7	0.5994	0.8362
16/2-14 T2	0.4206	0.7728
16/2-11 A	0.6734	0.7594
16/3-6	0.7776	0.8506
16/2-7 A	0.6469	0.7786
Total	0.6272	0.7984

For this model as well, it is interesting to investigate the prediction against the actual formations. Figure 5.10 illustrates how poor the model performed when prediction the

formations in well *16/2-14 T2*. From the very beginning, the model was unable to predict the formation, and as can be seen, some of the formations that were predicted were, in addition to being wrong, also in the wrong group. This is very interesting, considering the fact that the first formation present in well *16/2-14 T2* was the formation that was observed the most in the entire data set, and all the previously discussed models were able to predict this part with very high accuracy. Furthermore, we can see that the model struggled with close to all the formations in that particular well. The well in which the LSTM model obtained the highest accuracy, well *16/3-6*, depicts a model that is somewhat able to predict the formations, despite the fact that this model too struggles with the transitions between the formations and groups. We can see that contrary to well *16/2-14 T2*, the first part of the well was predicted perfectly, and a large section of in the middle of the well was also predicted with fairly high accuracy. In this well, most of the prediction errors that occurred were towards the bottom of the well. This is similar to what we observed in the other models and originates from, as previously discussed, the fact that the bottom part of the wells usually is more fragmented, compared to the top part of the wells.

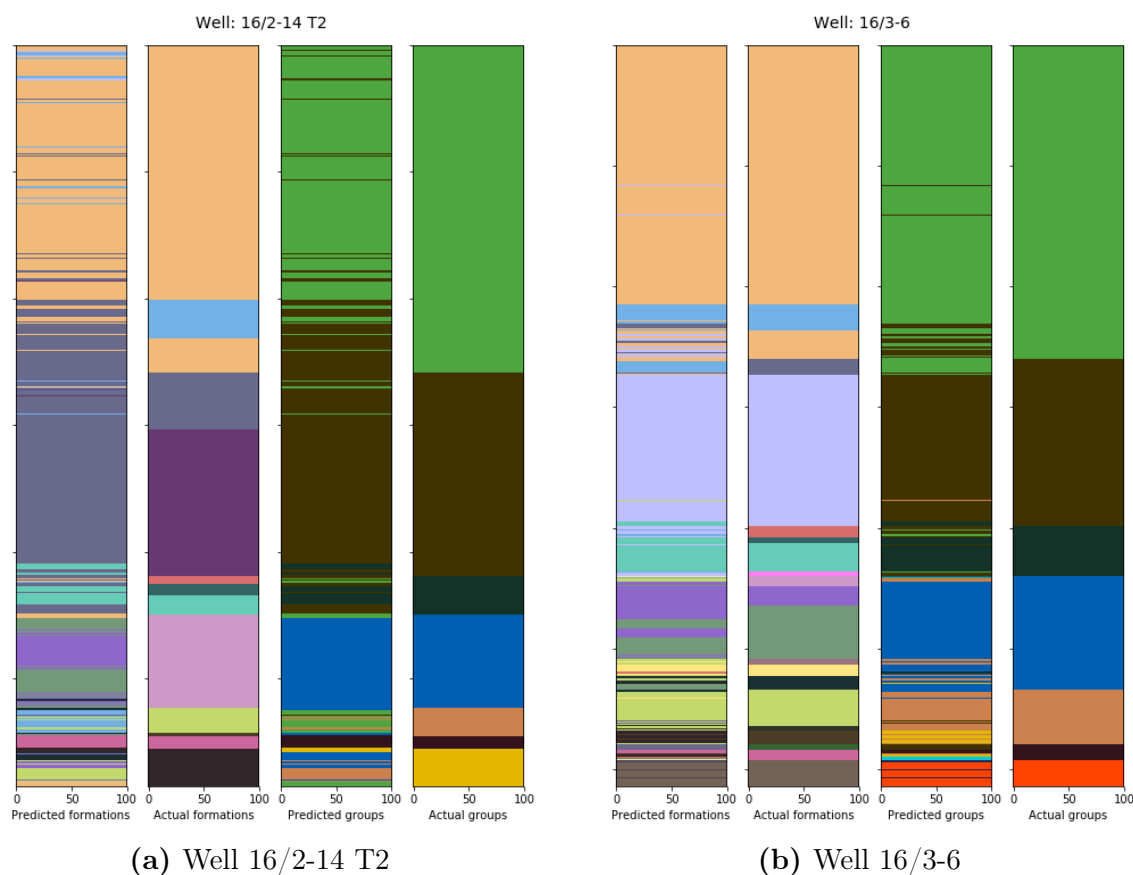


Figure 5.10: The LSTM model’s predicted formations and groups compared to the actual for the best and worse well. Each color represents a unique formation and group. The figure illustrates the areas in which the model predicted correctly and where the model was not able to make the right predictions.

5.5 Summary

In this analysis, we have examined how accurately the machine learning techniques *Logistic Regression*, *LightGBM*, and *LSTM* have been able to label wells with subsurface formations. To perform these predictions, the models were given different sets of wireline logs, recorded professionally by various companies in the drilling industry. The wells we have studied are located on the Norwegian Continental Shelf, or more specifically, in the Johan Sverdrup field. A complete comparison of the performance for the individual modes is shown in Table 5.9.

Table 5.9: Accuracy for all models on blind wells

	LogReg		LGBM 1		LGBM 2		LSTM	
	Formation	Group	Formation	Group	Formation	Group	Formation	Group
16/3-7	0.692	0.9312	0.7132	0.945	0.7091	0.9616	0.5994	0.8362
16/2-14 T2	0.5846	0.956	0.583	0.9153	0.5698	0.9451	0.4206	0.7728
16/2-11 A	0.6317	0.9384	0.8569	0.9472	0.8698	0.9651	0.6734	0.7594
16/3-6	0.7656	0.9062	0.8109	0.9332	0.8637	0.9559	0.7776	0.8506
16/2-7 A	0.8592	0.9415	0.888	0.9673	0.9184	0.9807	0.6469	0.7786
Total	0.7063	0.9347	0.7758	0.9423	0.7917	0.9621	0.6272	0.7984

The total accuracy illustrated in Table 5.9, shows that the models were able to pick up meaningful formation-patterns by combining information from different wireline logs. On a group-level, all models, except for LSTM, managed to predict formations placed in the same group remarkably accurate, with the best model yielding an average accuracy of 96%. However, on a formation-level, the differences between the models become more prominent, with total accuracies ranging from 63-79%. For all wells, the top part was predicted quite accurately, and because about 1/3 of the wells consisted of the same formation, this boosted the accuracy significantly. As we moved further down the wells, the models had more trouble differentiating between similar formations within the same group. In general, if we consider all the models' predictions for all blind wells (as illustrated in Appendix A5), there is a clear trend that the transition-points are the most difficult segments to predict, especially for the lower, more fragmented, part of the well.

To illustrate this trend, we have included a summary of all the models' predictions for well *16/2-7 A* in Figure 5.11. Figure 5.11, shows how most of the models were able to separate between similar formations with high accuracy. Further, it is very impressive to see how precise the LightGBM 2 model (c) was able to label the long- to mid-length sequences of the well, and even more so for the precise labeling of the transition-points between each formation for the lower, more fragmented part of the well.

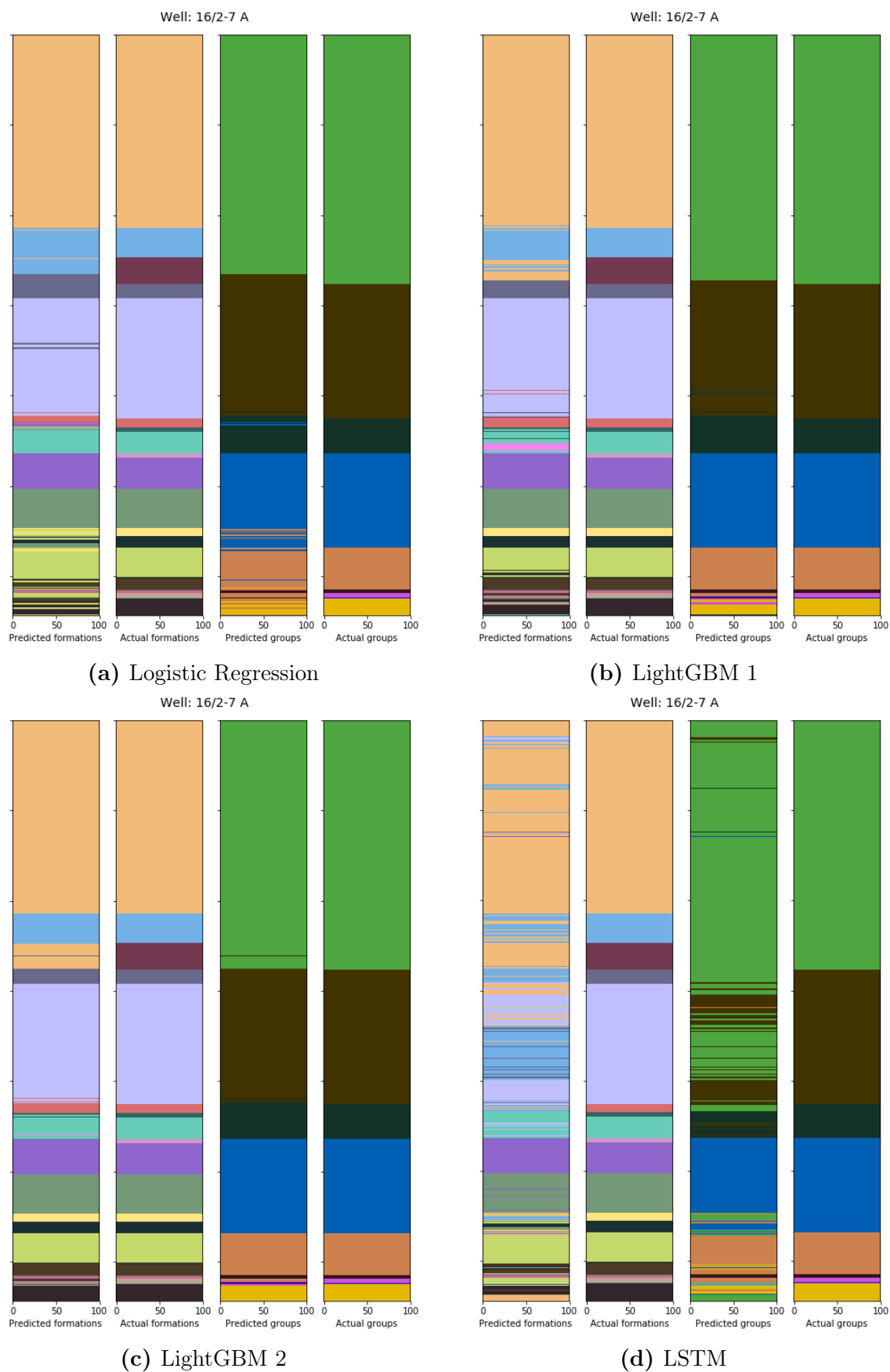


Figure 5.11: Well 16/2-7 A - Predicted vs actual formation and group for all models

To study how the features influenced our two best performing models to make its predictions, the package *SAHP* was utilized. We found that *true vertical depth* was, undoubtedly, the most important feature to determine which to predict. We found that the formation *undifferentiated nordland group* was, for both LightGBM models, characterized by a negative relationship with *true vertical depth*, and a positive relationship with the acoustic log measures (*dt*) and neutron porosity (*nphi*). In other words, when the observation was located at the top of our blind wells and had high values of acoustic and neutron porosity log, the model predicted the observation it to be *undifferentiated nordland group* with high probability. In addition, we found that both LightGBM models identified the formation *skade fm* as being location-specific, not only with respect to *true vertical depth*, but also the coordinates of the well. In short, the models gave the observed value a high probability of being *skade fm* if were found at a medium to high depth point, had low values of *dt*, *rdep*, and *rhob*, while being located at specific locations in the Johan Sverdrup field.

6 Discussion

Although the analysis showed that the application of machine learning could be helpful when classifying formations in the Johan Sverdrup field, there are some non-negligible limitations in the presented solution and some interesting areas for further research. In the thesis, we used a supervised learning approach, and the discussed models were able to predict the formations in the Johan Sverdrup field with relatively high accuracy, and the accuracy on the group level was even better. The supervised learning approach means that only the formations that the models have been exposed to can be predicted at a later point. This leads to an important point when working with machine learning; the availability of data and the quality of the available data. In the first part of this chapter, we will discuss the validity of our results before we elaborate further on the importance of domain knowledge. Lastly, some interesting suggestions for further research are presented.

6.1 Validity of Results

In terms of the validity of results, we note that throughout the entire process, the test set was kept separated from the train- and validation sets. As mentioned in Section 3.3.2, this was crucial in order to calculate the models' performance on unseen data and to compare the models to each other. Despite our effort to test the model as accurately as possible, it is important to highlight the fact that the test accuracy varies significantly within the individual wells in the test set. It is, therefore, possible that a different way of splitting the data could have yielded different results. This implies that the results are somewhat uncertain, and the degree of uncertainty of the models is an interesting area for further research.

Furthermore, the data itself played a major role in this thesis. As was discussed in Section 3.3.2, much time was spent making sure the data sets were representative of the strata in the Johan Sverdrup field in order for the models to be able to generalize well. We argued that this was important considering the fact that if there were formations in the test set that the model had not been trained on, the model would not be able to predict that formation.

The data set we utilized in the thesis was collected from the Norwegian Petroleum Directorate (NPD), which can be considered a valid source of information. However, as was seen in Section 4.1, more than half of the formations in the data set were either unlabeled or had no formal name. The data sets were reported to NPD from various sources and at different periods, meaning that the quality of the reported data might be of varying quality. The time aspect of the data collection is of relevance since the equipment being used has changed and improved throughout the years, and the fact that different companies have collected the information can also mean that differences in the data collection process, between the companies, can influence the results. Whereas a human, labeling the formations, can utilize the experience and potentially other types of measurements in order to identify the correct formations, the machine learning algorithms presented in this thesis are only able to make predictions based on the data given to it. In other words, if an algorithm is "fed" erroneous data, it will pick up false patterns, and with that, make incorrect classifications.

6.2 Domain Knowledge

In the present hype around machine learning, it can seem as if anyone can use machine learning, and no matter the task, the results will always be good. Companies are racing to implement machine learning in their everyday tasks as it is presented as the solution to all problems you might face.

In this hype, it can be easy to forget the importance of domain knowledge when working with machine learning. The problem statement at hand in this thesis can be viewed as a data science problem, but it is important to keep in mind that both the input- and output variables are geological variables. A deep understanding of how formations are formed, what values can be expected for the characteristics in certain formations, and other details related to the geology is definitely helpful when working with such a technical task. When we started working on the thesis, we had no previous knowledge of the geological aspect tackled in this thesis, and a lot of time was spent reading up on geological topics relevant to the problem (i.e., wireline logs and stratigraphy). In addition, we have collaborated closely with Pro Well Plan's experienced team of geologists, which have given us valuable support and suggestions throughout the whole process. Working on such a technical

geological task has definitely shown us the importance of solid domain knowledge as we would not have been able to obtain the results achieved in this thesis without the guidance from Pro Well Plan.

Furthermore, better domain knowledge opens up for the possibility to analyze the results in a more technical way by, for instance, investigating the wells in the train- and validation set in which the models had the weakest performance. By investigating the performance on individual wells further, it is potential for additional feature engineering and or a different tuning approach of the models, in order to have the models pick up on patterns were observed to be difficult to identify. A combination of solid domain knowledge and the interpretable machine learning solution presented can also lead to better understanding and potentially increase the predicting power further.

6.3 Further Research

The thesis opens up for several areas of further research. As was seen in Section 5.5, LightGBM 2, the model in which the train and validation set was stratified in the normal way, had the highest accuracy on the test set. The accuracy of **79.17%** was **1.59pp** above the second-best model, LightGBM 2. Despite the LightGBM 2 model achieving the highest accuracy, there are some problems related to the normal stratified split in our case, as was discussed in Section 3.3.2. During training and validation, the normal way of stratifying introduces information from the train set into the validation set. This is not necessarily a problem, as long as the model is able to generalize well on a blind test set. However, the fact that the model will tune the parameters using observations that are almost identical to observations in the train set can lead to serious overfitting. Overfitting refers to the production of an analysis that fails to reliably predict on unseen data since the analysis corresponds too closely or exactly to a particular data set (Lexico, nd). In our research, we observed that the best performing model, LightGBM 2, had an accuracy on the validation set equal to **99.96%**. When we see this in conjunction with the overall accuracy obtained on the test set, it is clear that the model has overfitted to the train and validation set. Furthermore, the fact that the validation accuracy is close to 100% also means that the model will not be able to learn further from the provided train and validation set, even with additional tuning efforts. A validation accuracy equal to

100% means that the model has no room for improvement and has, therefore, reached its maximum performance.

Contrary to LightGBM 2, the second-best performing model, LightGBM 1, did not overfit to the train- and validation set nearly as much. LightGBM 1 used data sets from our customized splitting function, which kept all sequential observations from the same well in the same data set. The fact that the observations in the wells were not split up meant the models could potentially pick up on important information in regards to the layers, as was discussed in Section 3.3.2. Furthermore, since the observations close to each other in a well are likely to contain similar values, our function avoided the leakage of information from the train set into the validation. The result was a model that obtained a validation accuracy was equal to **79.06%**. The validation accuracy was only **1.48pp** higher than the total accuracy on the blind wells, which indicates that LightGBM 1 did not overfit to the train- and validation set and still has a lot of potential for improvement through further tuning. It is, therefore, safe to say that the most promising model for further research is LightGBM 1, despite the fact that it had a lower test accuracy than LightGBM 2 on the blind wells.

In terms of the LSTM-model, we would like to emphasize the limited tuning efforts due to the time- and computational constraint. The number of epochs we were able to run and the fact that we were not able to experiment with the tuning of the hyperparameters as much as needed meant that it was not surprising that the model was unsuccessful. Through a better model tuning and the inclusion of the newly announced package [Keras Tuner](#), the results might have been different. Furthermore, as was seen in Section 5.5, the LSTM-network obtained an accuracy of **77.76%** on well *16/3-6*, an accuracy that was only slightly below some of the other models. This indicates that there is potential for improvement through additional tuning. As a result, despite the fact that the model had the weakest performance on this particular test set with the model tuning described in Section 5.4, we argue that it does not exclude it from further research.

6.3.1 Other Areas and Approaches

The code written for this project was written in a way that makes it highly reusable and therefore, applicable, to other oil-fields in which similar well logging data sets are

available. The re-usability was important through our entire project, and as a result, future analysis on other wells would be less time-consuming. By using the code available in the GitHub repository found in Appendix A1, a lot of the data wrangling and data cleaning will be automated, leaving more time for the analysis and tuning of the models. Since the research of the thesis was limited to the Johan Sverdrup field, another area of further research is the application of the models on wells in different fields. It would be interesting to compare the accuracy of the predictions obtained in this thesis to the accuracy obtained in other areas.

Furthermore, the analysis of our two best performing models, LightGBM 1 and LightGBM 2, in Section 5, showed that *true vertical depth* undoubtedly was the most important feature when predicting formations. To maximize the accuracy of our predictions, we decided to keep this variable as an input feature for all experiments performed. However, for further research, it would be interesting to see if the models would be able to pick up other patterns with regards to formation characteristics, by only taking features related to rock-properties (e.g., wireline logs) as input.

6.3.2 Transfer Learning

Even though the performance of the LSTM model was significantly worse, than the other models presented in the thesis, the neural network has an advantage over the other in terms of weight sharing. Weight sharing, also known as transfer learning, has become increasingly popular within the field of deep learning and describes the process of utilizing the weights of a network that has been trained on a large data set (Chollet, 2018). An interesting approach would be to train the model on the full data set, consisting of all 614 wells spread across the Norwegian Continental Shelf, and then use the weights to further train the model on individual fields. If the original data set is large- and general enough, the pre-trained network can in, effect, be used as a generic model and its features could be used for other classification problems, even if the new problems contained different classes than the classes available in the original data set (Chollet, 2018). As an example, a neural network trained to classify formations could, through transfer learning, add value to a network tasked with classifying formations in different locations or even a different classification problem altogether, for instance, facies.

6.3.3 Model Ensembling

Another approach that can be interesting to investigate further is model ensembling. Model ensembling is a technique in which the predictions of different sets of models are pooled together in order to improve the final prediction. It relies on the assumption that different models trained independently will look at different aspects of the data, which in turn will lead to a more accurate description of the data (Chollet, 2018). By, for instance, combining the predictions from the models in the thesis through a weighted average, where the best performing classifiers would be given a higher weight than the worse performing classifiers, we could have utilized the diversity in the models. Additionally, we could have used model stacking, a technique in which the predictions from one or more models are fed to a different model.

Further research on the effect of model ensembling in the field of formation classification, using wireline logs as input, could potentially lead to increased predicting power and possibly better results than what was achieved in this thesis. This is further backed by the fact that the LSTM model had the highest accuracy on well *16/3-6*, whereas all the other models achieved the highest accuracy on well *16/2-7 A*. This suggests that the models looked at different aspects of the data set, and combined, they could potentially have yielded better predictions on the test set.

6.4 The Potential of Machine Learning in Stratigraphy Classification

As discussed in the introduction, a more streamlined and data-driven approach to stratigraphy interpretation can have a huge economic impact on the drilling companies involved. By liberating the experts from their manual task of stratigraphy interpretation, they can focus their time and resources on other tasks that can further improve the well design. An automatic stratigraphy interpretation can help lower the risk of downtime and failure, and thereby potentially saving the companies millions (M. Tvedt, personal communication, December 17, 2019).

In this thesis, we found that machine learning can, with relatively high accuracy, label wells

in the Johan Sverdrup field with subsurface formations. However, the results varied quite a lot between the individual wells in our test set, even for our two best performing models, LightGBM 1 and LightGBM 2. This indicates that models would require additional work before they could generalize better on unseen data, and be fully integrated into the well design process. On the other hand, by utilizing the proposed algorithms, the geologists can get clear indications on what the models have predicted, both with respect to performance and its way of "thinking". This means that the geologists can focus their time and resources on the formations in which the algorithms are less certain of, freeing up time for deeper analysis into the more difficult cases. In addition, we found that the interpretable machine learning solution presented in Section 3.4 can be of great assistance when analyzing and finding geological patterns for the difficult formations.

7 Conclusion

The purpose of this thesis was to investigate to what extent machine learning techniques can use wireline logs as input in order to label subsurface formations. To achieve this, we compared four different experimental approaches with three different machine learning algorithms. The models were trained-, validated-, and tested on wells in the Johan Sverdrup field. The models were compared using an accuracy measure on a blind test set, seen in conjunction with a plot in which the predicted values were compared to the actual values. Furthermore, Shapley Additive Explanations (SHAP) was used as an interpretable machine learning approach in order to better understand how the LightGBM models made their predictions.

The main conclusion that can be drawn from the thesis is that sophisticated machine learning techniques are able to classify subsurface formations with relatively high accuracy. The two best performing models were both built using a LightGBM algorithm, but with two different approaches to the training- and validation split. The model that yielded the highest accuracy was the model identified as LightGBM 2, with an accuracy of 79.17%. The accuracy was 1.59pp higher than the accuracy of the model identified as LightGBM 1, which had an accuracy of 77.58%. Despite this, we concluded that LightGBM 1 had the highest potential for further research of the models compared in the thesis. Due to severe overfitting to training- and validation data in LightGBM 2, we argued that the model not would be able to learn further and thereby not be able to improve the accuracy of the predictions. LightGBM 1, on the other hand, did not overfit nearly as much, which indicates that additional tuning can potentially increase the accuracy further.

The main contribution of the thesis can be split into three parts. Firstly, our research showed that machine learning could predict subsurface formations in the Johan Sverdrup field with an average accuracy of around 80%. Secondly, we showed how the interpretable machine learning method *SHAP* can be used to gain a deeper understanding of how models utilize information from wireline logs to make its predictions. These two findings can be of great assistance to engineers and geologists currently performing the labeling of the formations in a manual fashion. By utilizing the classification methods described, experts can spend more time on other value-creating tasks. Lastly, the code written

for the project is generalized, meaning it can be applied to any data set with a similar structure. For further work, we proposed tuning the models further and testing other machine learning techniques in addition to investigating the models' ability to classify formations in other areas.

References

- Ahmadi, M. A., Soleimani, R., Lee, M., Kashiwao, T., & Bahadori, A. (2015). Determination of oil well production performance using artificial neural network (ann) linked to the particle swarm optimization (psa) tool. *Petroleum*, 1(2), 118–132.
- Ali Amin-Nejad (2019). Complex dataset split - stratifiedgroupshufflesplit. Retrieved 02 November, 2019, from <https://stackoverflow.com/questions/56872664/complex-dataset-split-stratifiedgroupshufflesplit>.
- Almaliki, Z. A. (2018). Bias-variance dilemma? Retrieved 29 November, 2019, from <https://towardsdatascience.com/bias-variance-dilemma-74e5f1f52b12>.
- Andresen, P. A. (2019). Digital well delivery requires contextualization. Retrieved 20 November, 2019, from <https://www.offshore-mag.com/drilling-completion/article/14069059/digital-well-delivery-requires-contextualization>.
- Asaithambi, S. (2017). Why, how and when to scale your features. Retrieved 15 November, 2019, from <https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e>.
- Asquith, G., Krygowski, D., Henderson, S., & Hurley, N. (2004). *Basic well log analysis*. American Association of Petroleum Geologists.
- Ayala, L. F. & Ertekin, T. (2005). Analysis of gas-cycling performance in gas/condensate reservoirs using neuro-simulation, 10.
- Bissuel, A. (2019). Hyper-parameter optimization algorithms: a short review. Retrieved 20 November, 2019, from <https://medium.com/criteo-labs/hyper-parameter-optimization-algorithms-2fe447525903>.
- Bøe, H. W. (2018). Enhanced lithological description of the jurassic sequence in the viking graben and oseberg terrace using machine learning. Master's thesis, NTNU.
- Bølviken, E., Storvik, G., Nilsen, D. E., Siring, E., & Van Der Wel, D. (1992). Automated prediction of sedimentary facies from wireline logs. *Geological Society, London, Special Publications*, 65(1), 123–139.
- Brookshire, B. (2018). Scientists say: Stratigraphy. Retrieved 02 December, 2019, from <https://www.sciencenewsforstudents.org/blog/scientists-say/scientists-say-stratigraphy>.
- Brownlee, J. (2017a). A gentle introduction to exploding gradients in neural networks. Retrieved 02 November, 2019, from <https://machinelearningmastery.com/exploding-gradients-in-neural-networks/>.
- Brownlee, J. (2017b). Gentle introduction to the adam optimization algorithm for deep learning. Retrieved 23 November, 2019, from <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>.
- Brownlee, J. (2019a). Difference between a batch and an epoch in a neural network. Retrieved 27 November, 2019, from <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/>.

- Brownlee, J. (2019b). A gentle introduction to dropout for regularizing deep neural networks. Retrieved 27 November, 2019, from <https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/>.
- Brownlee, J. (2019c). Gentle introduction to the bias-variance trade-off in machine learning. Retrieved 03 December, 2019, from <https://machinelearningmastery.com/gentle-introduction-to-the-bias-variance-trade-off-in-machine-learning/>.
- Brownlee, J. (2019d). Understand the impact of learning rate on neural network performance. Retrieved 25 November, 2019, from <https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>.
- Bushaev, V. (2018). Understanding rmsprop - faster neural network learning. Retrieved 25 November, 2019, from <https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a>.
- Bérard, T., Chugunov, N., Desroches, J., & Prioul, R. (2019). Feasibility and design of hydraulic fracturing stress tests using a quantitative risk assessment and control approach. *Petrophysics – The SPWLA Journal of Formation Evaluation and Reservoir Description*, 60(1), 113–135.
- Chollet, F. (2018). *Deep learning with Python*. Manning Publications Co.
- Chollet, F. & Allaire, J. J. (2018). *Deep Learning with R* (1st ed.). Greenwich, CT, USA: Manning Publications Co.
- Collier, A. (2015). Making sense of logarithmic loss. Retrieved 03 December, 2019, from <https://www.r-bloggers.com/making-sense-of-logarithmic-loss/>.
- Crooks, E. (2018). Drillers turn to big data in the hunt for more, cheaper oil. Retrieved 28 October, 2019, from <https://www.ft.com/content/19234982-0cbb-11e8-8eb7-42f857ea9f09>.
- Czypionka, R. D., Gulewicz, D., Keith, D., & Rey, M. (2016). Reservoir facies impact on drilling, completion and production in the cardium tight oil play. In *AAPG Annual Convention and Exhibition*.
- De Mulder, W., Bethard, S., & Moens, M.-F. (2014). A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech Language*, 30.
- Directorate, N. P. (2019). Wellbore - factpages - npd. Retrieved 05 December, 2019, from <https://factpages.npd.no/factpages/Default.aspx?culture=en&nav1=wellbore&nav2=Attributes>.
- Donges, N. (2018). Recurrent neural networks and lstm. Retrieved 03 November, 2019, from <https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5>.
- Dubois, M. K., Bohling, G. C., & Chakrabarti, S. (2007). Comparison of four approaches to a rock facies classification problem. *Computers and Geosciences*, 33(5), 599–617.
- Equinor (2019). Johan sverdrup, the north sea giant, is on stream. Retrieved 17 December, 2019, from <https://www.equinor.com/en/news/2019-10-johan-sverdrup.html>.

- Fu, Z. (2016). Comparison of gradient descent, stochastic gradient descent and l-bfgs. Retrieved 27 November, 2019, from <http://www.fuzihao.org/blog/2016/01/16/Comparison-of-Gradient-Descent-Stochastic-Gradient-Descent-and-L-BFGS/>.
- Geological Survey of Norway (2015). Stratigraphy. Retrieved 04 December, 2019, from <https://www.ngu.no/en/topic/stratigraphy>.
- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.
- Glover, P. (2014). Petrophysics msc course notes. Retrieved 15 October, 2019, from http://homepages.see.leeds.ac.uk/~earpwjg/PG_EN/CDCContents/GGL-66565PetrophysicsEnglish/Chapter11.PDF.
- Grootendorst, M. (2019). Validating your machine learning model. Retrieved 20 November, 2019, from <https://towardsdatascience.com/validating-your-machine-learning-model-25b4c8643fb7>.
- Gupta, P. (2017). Regularization in machine learning. Retrieved 02 December, 2019, from <https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a>.
- Gómez, R. (2018). Understanding categorical cross-entropy loss, binary cross-entropy loss, softmax loss, logistic loss, focal loss and all those confusing names. Retrieved 23 November, 2019, from https://gombru.github.io/2018/05/23/cross_entropy_loss/.
- Hall, B. (2016). Facies classification using machine learning. *The Leading Edge*, 35(10), 906–909.
- Hall, B. (2019). Developments in machine and deep learning for facies classification. Retrieved 27 November, 2019, from <https://www.linkedin.com/pulse/developments-machine-deep-learning-facies-brendon-hall/>.
- Hall, M. & Hall, B. (2017). Distributed collaborative prediction: Results of the machine learning contest. *The Leading Edge*, 36(3), 267–269.
- Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780.
- Imamverdiyev, Y. & Sukhostat, L. (2019). Lithological facies classification using deep convolutional neural network. *Journal of Petroleum Science and Engineering*, 174, 216–228.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning – with Applications in R*, volume 103 of *Springer Texts in Statistics*. New York: Springer.
- Kapil, D. (2019). Hyperparameter search: Bayesian optimization. Retrieved 21 November, 2019, from <https://medium.com/analytics-vidhya/hyperparameter-search-bayesian-optimization-14be6fbb0e09>.
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 3146–3154). Curran Associates, Inc.

- Koehrsen, W. (2018). A conceptual explanation of bayesian hyperparameter optimization for machine learning. Retrieved 20 November, 2019, from <https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f>.
- Kurita, K. (2018). Lightgbm and xgboost explained. Retrieved 10 December, 2019, from <https://mlexplained.com/2018/01/05/lightgbm-and-xgboost-explained/>.
- Lasscock, B. (2019). Using deep learning to predict well-log facies. Retrieved 06 November, 2019, from <https://www.linkedin.com/pulse/using-deep-learning-predict-well-log-facies-ben-lasscock/>.
- Lexico (n.d). Overfitting. Retrieved 18 December, 2019, from <https://www.lexico.com/definition/overfitting>.
- Leyland, L. (2017). The importance and applications of well logging. Retrieved 27 October, 2019, from <https://medium.com/@welllogging/the-importance-and-applications-of-well-logging-3fe97fd1fa0e>.
- Lundberg, S. (2017). A game theoretic approach to explain the output of any machine learning model. Retrieved 13 December, 2019, from <https://github.com/slundberg/shap?fbclid=IwAR0BANPPafSR0-HjHoK5bd-UV8keczQo2PmQG76cbPAdJ5lTnDVTues2v6k>.
- Lundberg, S. M. & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 4765–4774). Curran Associates, Inc.
- Molnar, C. (2019). *Interpretable Machine Learning*. <https://christophm.github.io/interpretable-ml-book/>.
- Nasralla, S. (2018). Digitalization can save oil upstream business \$73 billion a year: Woodmac. Retrieved 04 November, 2019, from <https://www.reuters.com/article/us-oil-digital-savings/digitalization-can-save-oil-upstream-business-73-billion-a-year-woodmac-idUSKCN1NH0QR>.
- Nguyen, M. (2018). Illustrated guide to lstm’s and gru’s: A step by step explanation. Retrieved 02 November, 2019, from <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21/>.
- Norwegian Petroleum Directorate (n.da). Npd guidelines for designation of wells and wellbores. Retrieved 01 December, 2019, from <https://www.npd.no/globalassets/1-npd/regelverk/tematiske-veiledninger/eng/guidelines-for-designation-of-wells-and-wellbores.pdf>.
- Norwegian Petroleum Directorate (n.db). Well classification. Retrieved 01 December, 2019, from <https://www.npd.no/en/facts/wells/well-classification/>.
- of Encyclopaedia Britannica, T. E. (2014). Stratification. Retrieved 15 November, 2019, from <https://www.britannica.com/science/stratification-geology>.
- Olah, C. (2015). Understanding lstm networks. Retrieved 30 November, 2019, from <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.

- Olejnik, J., Karaffa, M. D., & Fleming, A. H. (n.d). Gamma-ray logs. Retrieved 20 October, 2019, from <https://igws.indiana.edu/AllenCounty/gammaRay>.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct), 2825–2830.
- Rey, J., Galeotti, S., et al. (2008). *Stratigraphy: terminology and practice*. Editions Technip.
- Rider, M. (1991). The geological interpretation of well logs.
- Rider, M. (2011). The geological interpretation of well logs.
- Roth, A. E. (1988). *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press.
- Salvador, A. & Murphy, M. A. (1998). International ttratigraphic guide — an abridged version. *Episodes.*, 22(4), 255.
- SHAP-Developers (n.d). Shap (shapley additive explanations). Retrieved 07 December, 2019, from <https://shap.readthedocs.io/en/latest/#>.
- Shi, H. (2007). Best-first decision tree learning. Master's thesis, The University of Waikato.
- Shulga, D. (2018). 5 reasons "logistic regression" should be the first thing you learn when becoming a data scientist. Retrieved 04 December, 2019, from <https://towardsdatascience.com/5-reasons-logistic-regression-should-be-the-first-thing-you-learn-when-become-a-data-scientist-fcaae46605c4>.
- Singh, S. (2018). Understanding the bias-variance tradeoff. Retrieved 29 November, 2019, from <https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229>.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1), 1929–1958.
- Stark, G. (2018). Gamma ray. Retrieved 19 November, 2019, from <https://www.britannica.com/science/gamma-ray>.
- Steingrimsson, B. et al. (2011). Geothermal well logging: Geological wireline logs and fracture imaging. *Short Course on Geothermal Drilling, Resource Development and Power Plants, El Salvador*.
- V, A. S. (2017). Understanding activation functions in neural networks. Retrieved 25 November, 2019, from <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>.
- Vakarelov, B. (2016). The "art" of well log correlation: practical tips and other musings. Retrieved 27 October, 2019, from <https://www.linkedin.com/pulse/art-well-log-correlation-practical-tips-other-musings-boyan-vakarelov/>.
- Van Rossum, G. & Drake Jr, F. L. (1995). *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.

- Verma, S. (2019). Understanding input and output shapes in lstm: Keras. Retrieved 24 November, 2019, from <https://medium.com/@shivajbd/understanding-input-and-output-shape-in-lstm-keras-c501ee95c65e>.
- Wang, X., Yang, S., Zhao, Y., & Wang, Y. (2018). Improved pore structure prediction based on micp with a data mining and machine learning system approach in mesozoic strata of gaoqing field, jiyang depression. *Journal of Petroleum Science and Engineering*, *171*, 362–393.
- Weller, M. (2018). Recurrent neural networks for time series forecasting. Retrieved 02 November, 2019, from <https://www.novatec-gmbh.de/en/blog/recurrent-neural-networks-for-time-series-forecasting/>.
- Wells, P. (2019). What you need to know about digital wells and field optimization. Retrieved 18 October, 2019, from <https://www.arundo.com/thejourney/what-you-need-to-know-about-digital-wells-and-field-optimization>.
- Yan, S. (2016). Understanding lstm and its diagrams. Retrieved 05 October, 2019, from <https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714>.
- Zazoun, R. S. (2013). Fracture density estimation from core and conventional well logs data using artificial neural networks: The cambro-ordovician reservoir of mesdar oil field, algeria. *Journal of African Earth Sciences*, *83*, 55–73.

Appendix

A1 GitHub Repository

Link to GitHub repository:

[pro-well-plan/thesis_stratigraphy_prediction_2019](https://github.com/pro-well-plan/thesis_stratigraphy_prediction_2019)

A2 Data

Table A2.1: Missing values for variables in original data set

Variable	Missing values
depth	0
md	0
main_area	0
long	0
lat	0
title	0
tvd	0
gr	0
formation	0
field	0
group	0
rmed	4711
rdep	4748
dt	138284
nphi	200460
rhob	200804
dts	214110
rsh	342392
hgr	342392
hrhob	342392
hrdep	342392
hnphi	342392
hrsh	342392
hrmed	342392

A3 Crossplot matrix

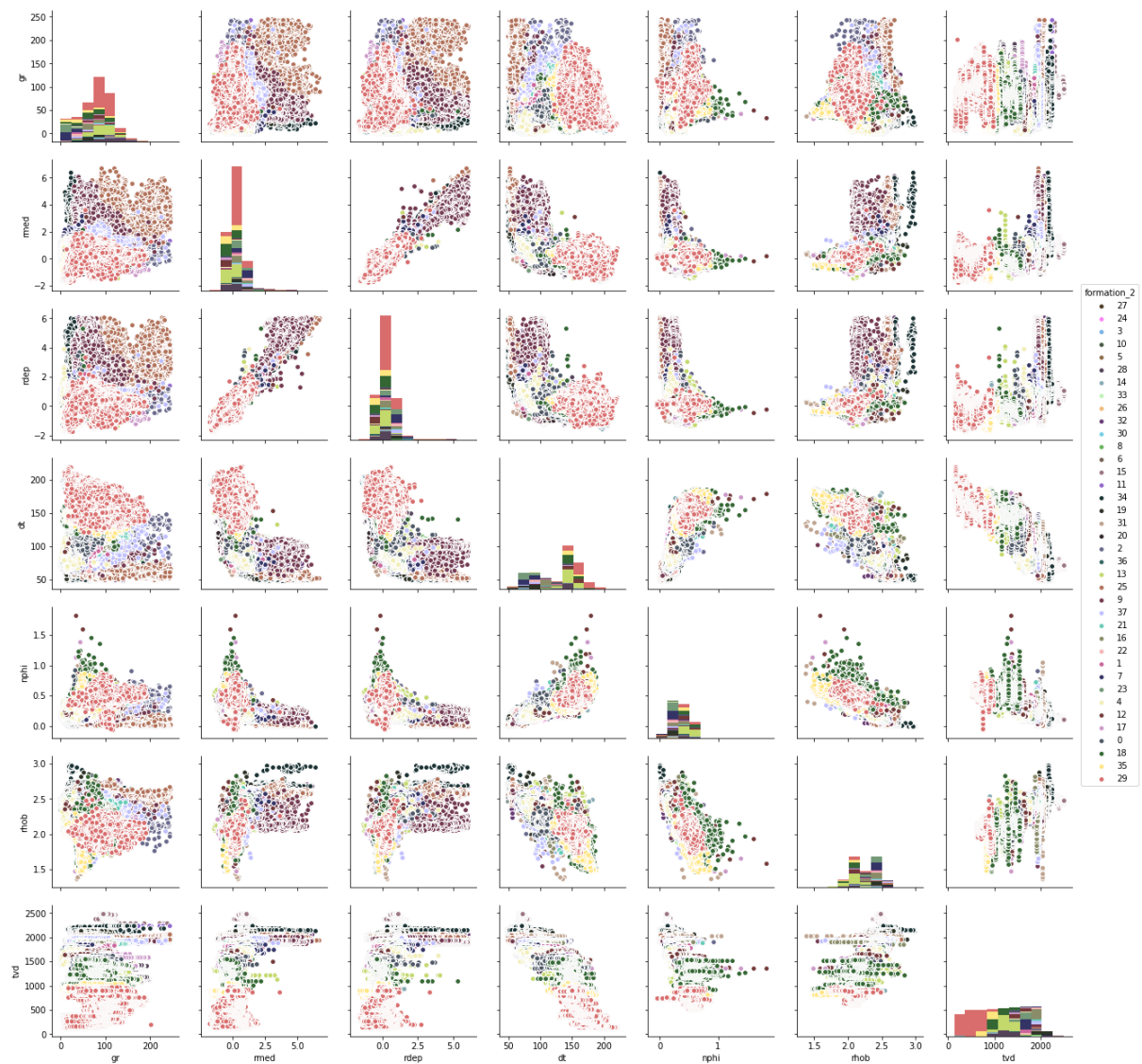


Figure A3.1: Crossplot matrix of the formations in the data set

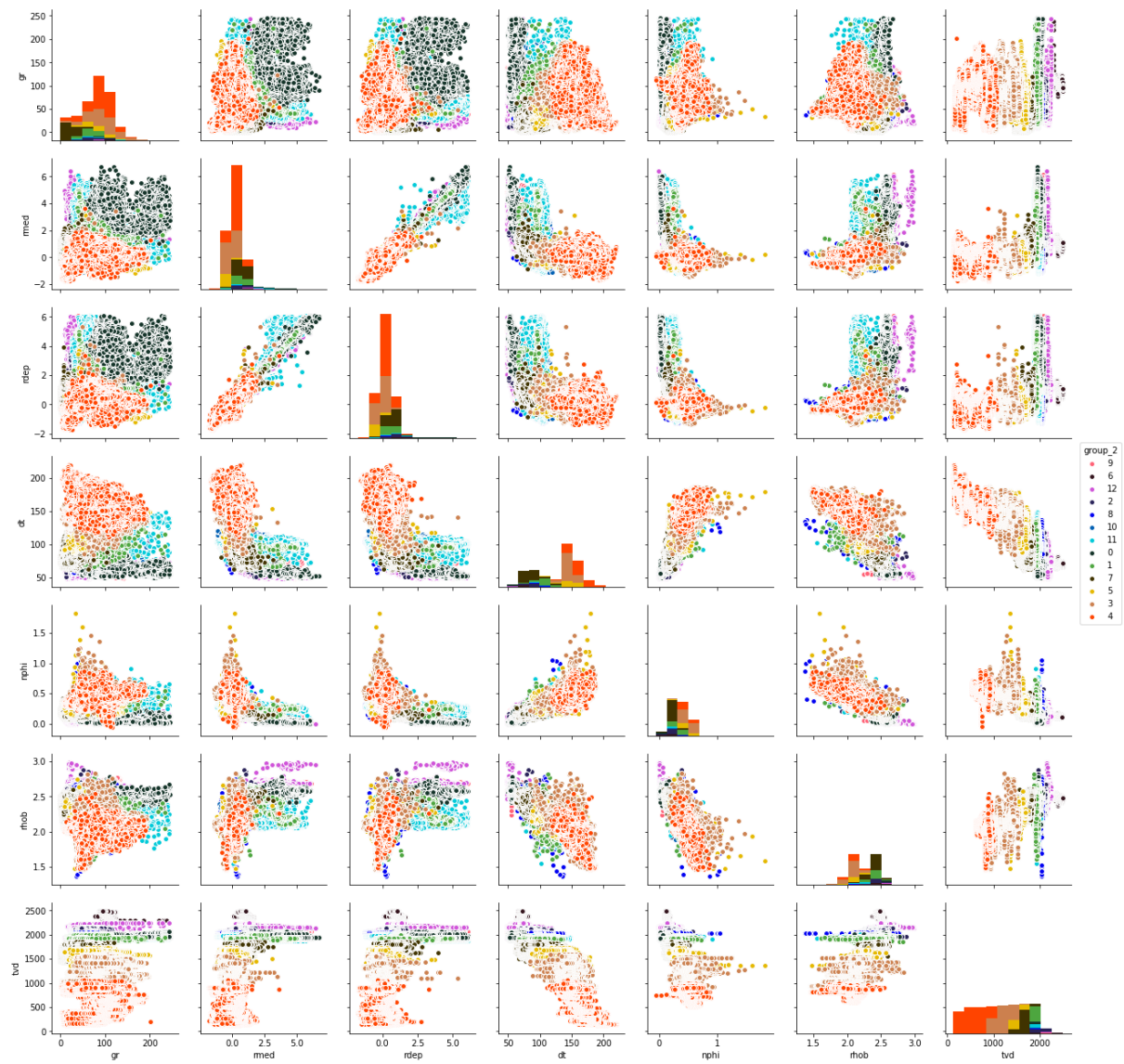


Figure A3.2: Crossplot matrix of the groups in the data set

A4 Boxplots

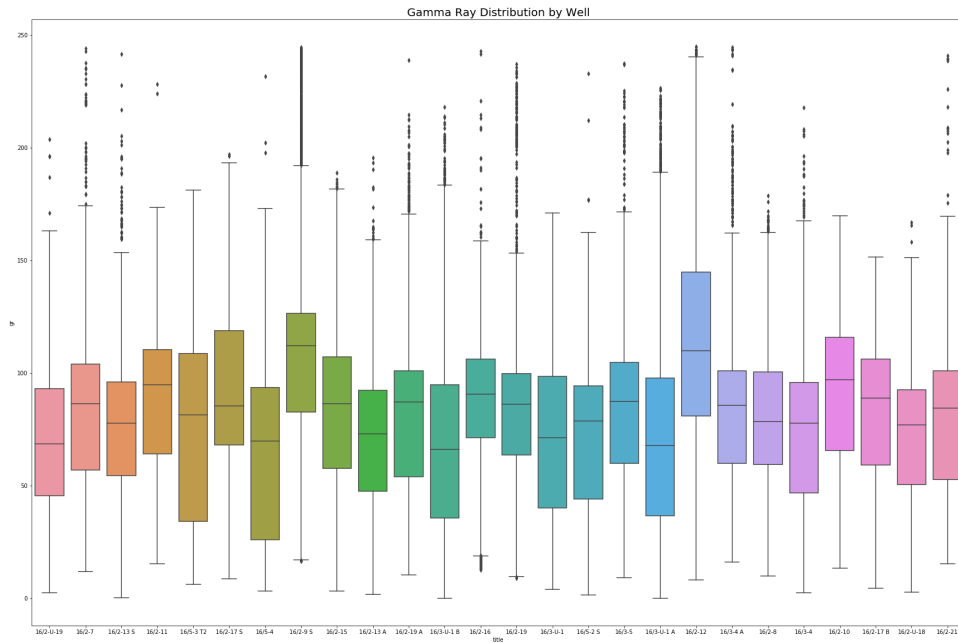


Figure A4.1: Boxplot of the gamma ray separated by well

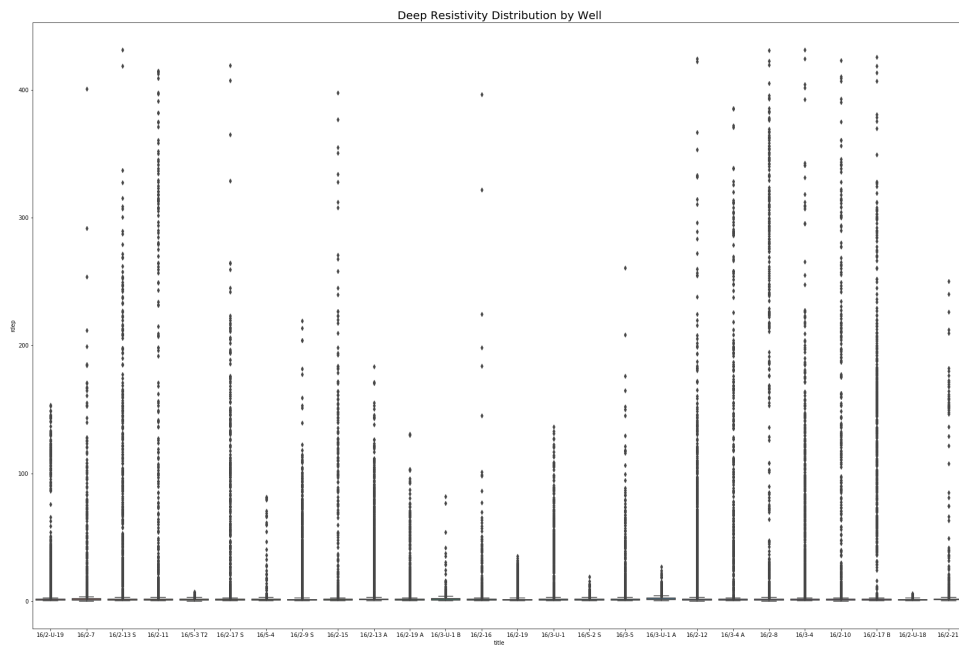


Figure A4.2: Boxplot of the deep resistivity separated by well

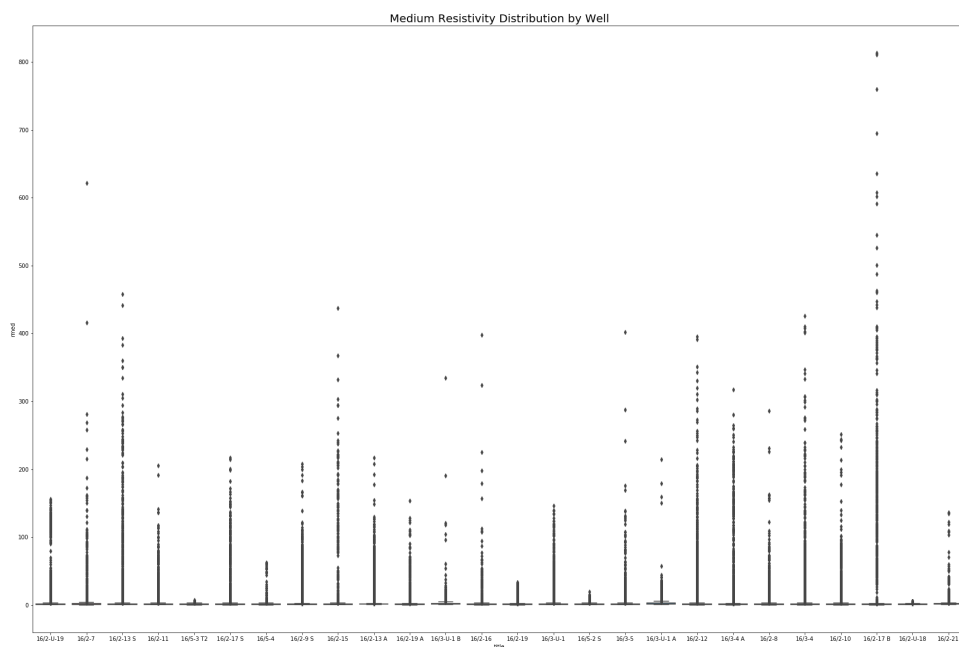


Figure A4.3: Boxplot of the medium resistivity separated by well

A5 Predicted vs. Actual

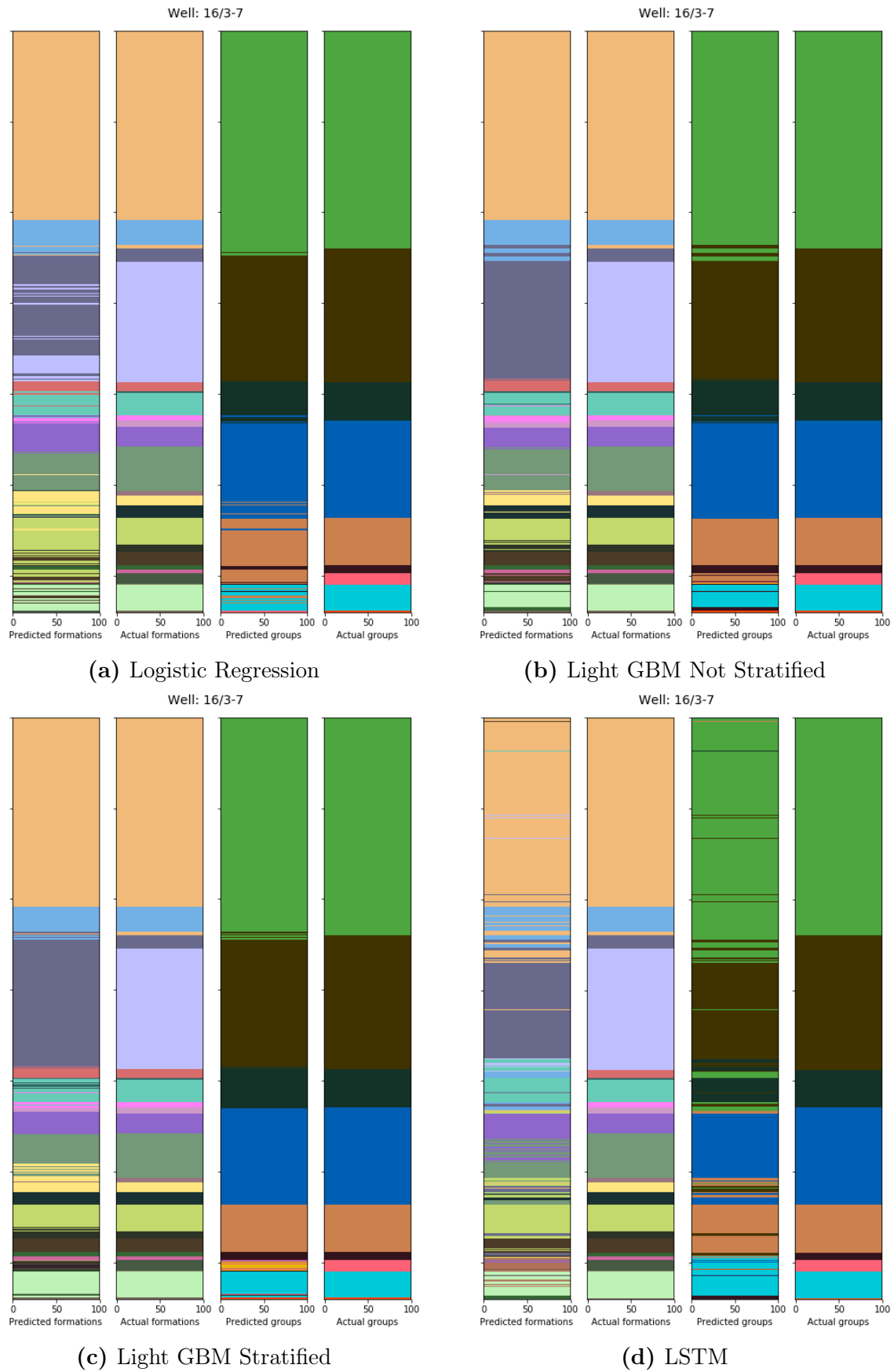


Figure A5.1: Well 16/3-7 - Predicted vs actual formation and group for all models

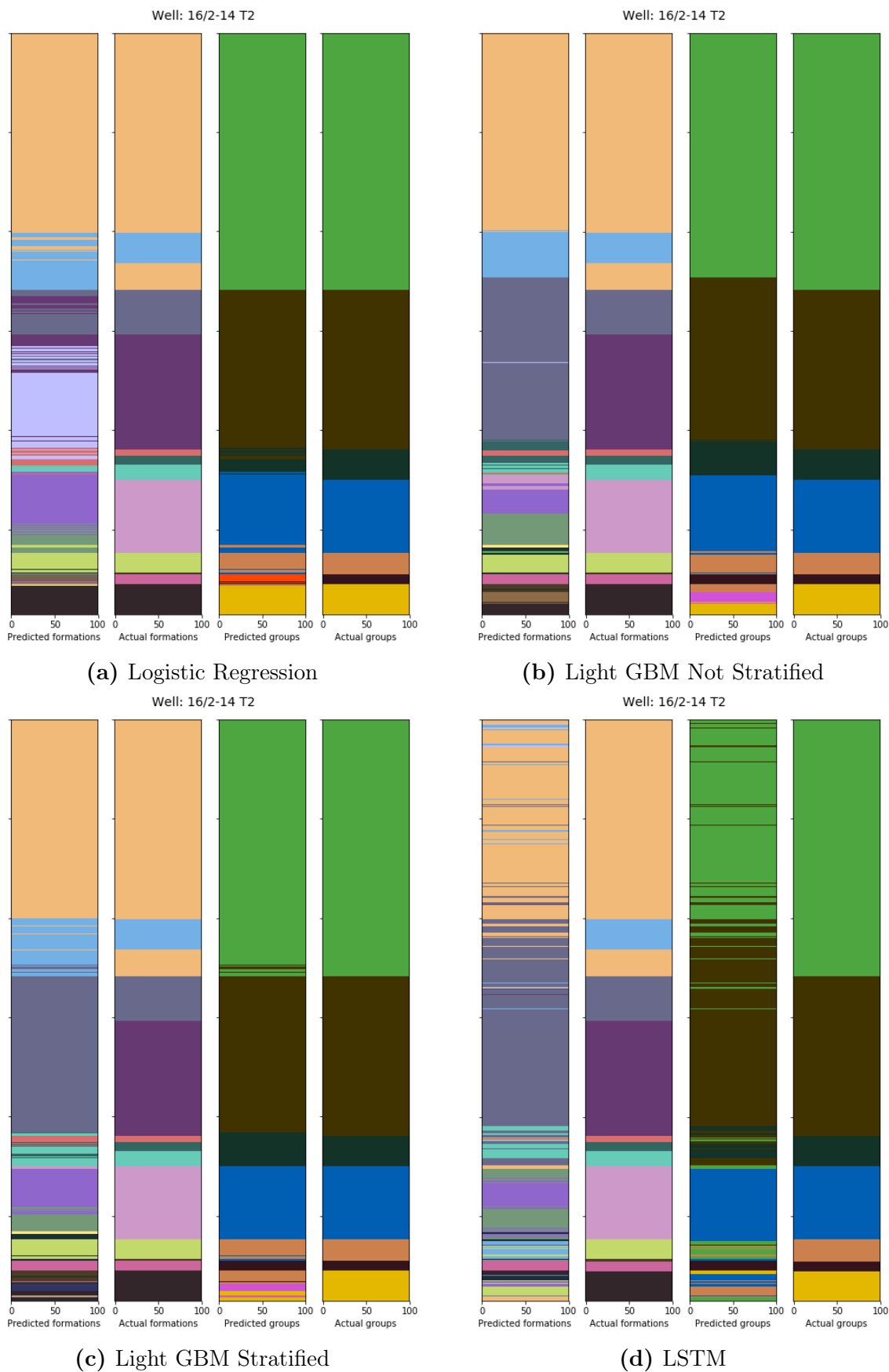


Figure A5.2: Well 16/2-14 T2 - Predicted vs actual formation and group for all models

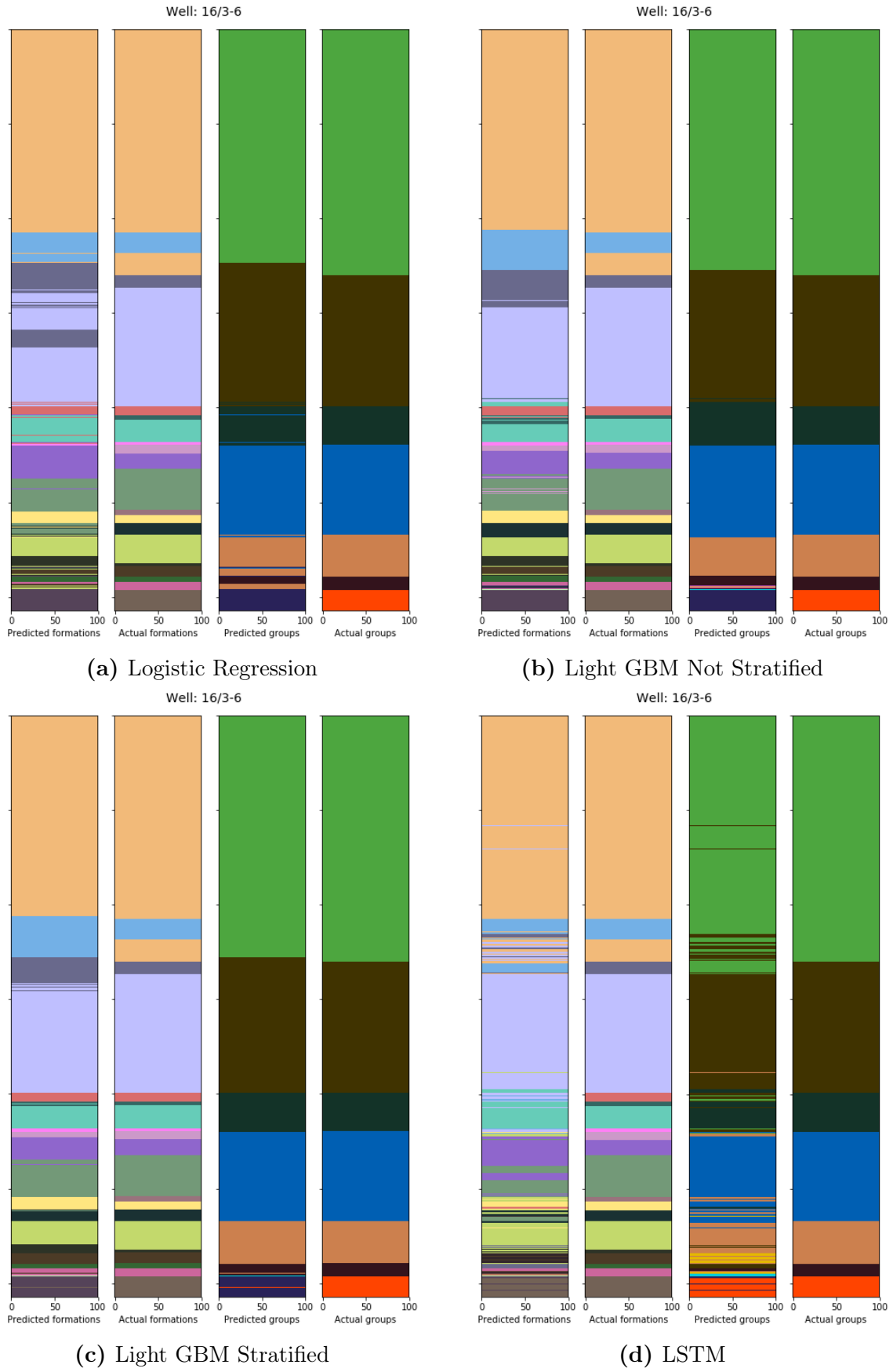


Figure A5.3: Well 16/3-6 A - Predicted vs actual formation and group for all models

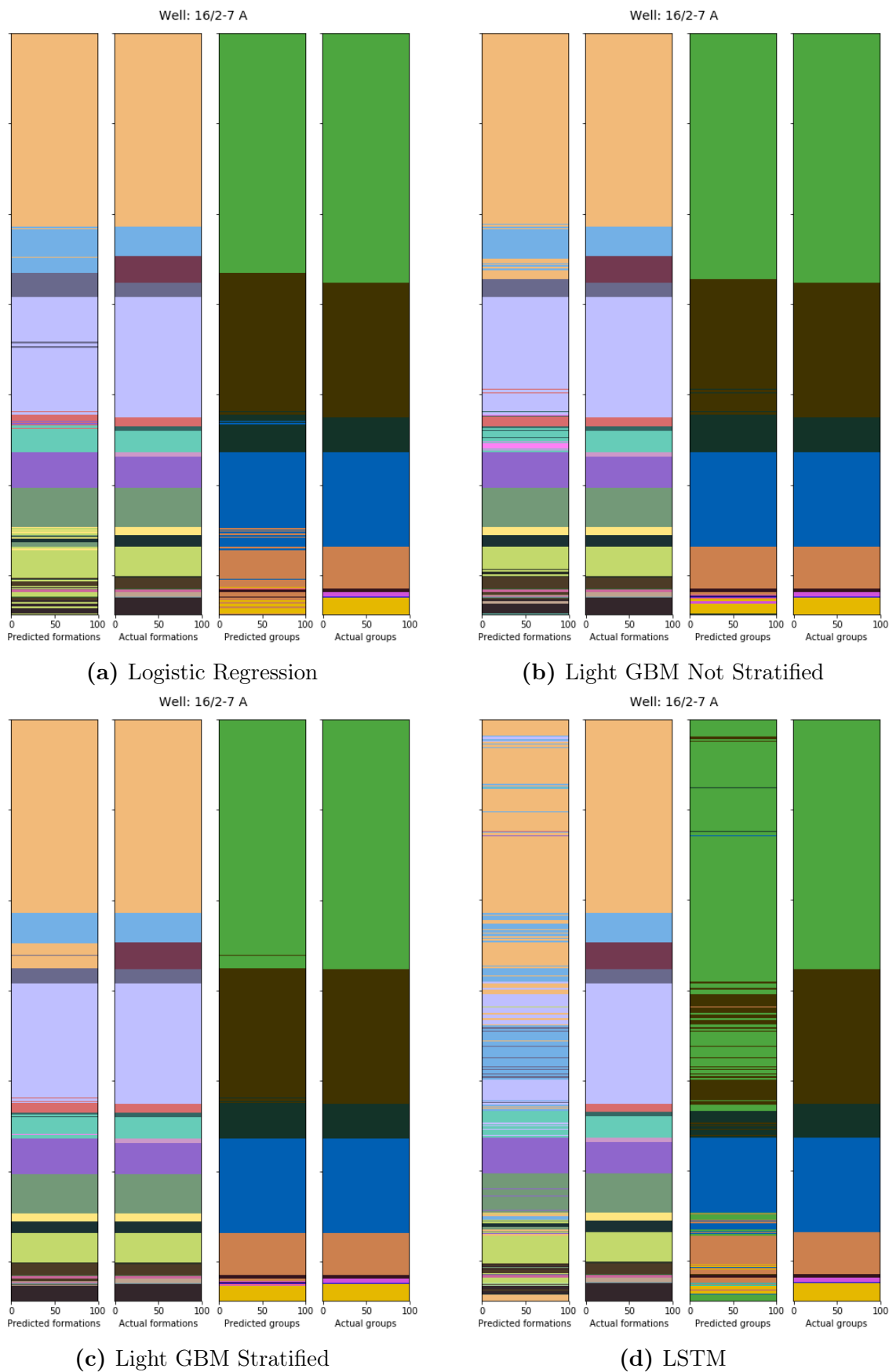


Figure A5.4: Well 16/2-7 A - Predicted vs actual formation and group for all models

A6 Original name and corresponding numerical values

A6.1 Groups

Table A6.1: Group dictionary

Group name	Group number
basement	0
cromer knoll gp	1
hegre gp	2
hordaland gp	3
nordland gp	4
rogaland gp	5
rotliegend gp	6
shetland gp	7
statfjord gp	8
undefined gp	9
vestland gp	10
viking gp	11
zechstein gp	12

A6.2 Formations

Table A6.2: Formation dictionary

Formation name	Formation number
balder fm	0
blodøks fm	1
draupne fm	2
eiriksson fm	3
ekofisk fm	4
grid fm	5
heather fm	6
hod fm	7
hugin fm	8
intra draupne fm ss	9
intra heather fm ss	10
kupferschiefer fm	11
lista fm	12
no formal name_hordaland gp	13
no formal name_nordland gp	14
no formal name_rotliegend gp	15
rødby fm	16
sele fm	17
skade fm	18
skagerrak fm	19
sleipner fm	20
sola fm	21
svarte fm	22
tor fm	23
tryggvason fm	24
undifferentiated_basement	25
undifferentiated_cromer knoll gp	26
undifferentiated_hegre gp	27
undifferentiated_hordaland gp	28
undifferentiated_nordland gp	29
undifferentiated_rotliegend gp	30
undifferentiated_statfjord gp	31
undifferentiated_undefined gp	32
undifferentiated_vestland gp	33
undifferentiated_zechstein gp	34
utsira fm	35
våle fm	36
åsgard fm	37