Universitetet i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program / Specialization Petroleum Geosciences Engineering	Spring semester, 2021 Open/ Confidential
Author: Benjamin Chinedu Nelson	…….. (Signature of the author)
Supervisor: Arild Buland	
Thesis Title: A study of machine learning models application for porosity prediction using petrophysical well logs. Case Study: The Brent Group – Statfjord field	
Credits (ECTS): 30	
Keywords Machine learning; Porosity prediction; Well logs; Regression; Performance evaluation	Number of pages: 68 + enclosures: 22 Stavanger, June 11th 2021

Copyright

A study of machine learning models application for porosity prediction using petrophysical well logs. Case Study: The Brent Group – Statfjord field

Author

Benjamin Chinedu Nelson

MSc. Thesis

Presented to the Faculty of Science and Technology

University of Stavanger

2021

Acknowledgements

First and foremost, this thesis is dedicated to my loving father, who encouraged me to pursue my master's degree. He started with me on this journey but could not make it to watch me complete it.

I would especially want to extend my appreciation to my wife, mother, sisters, and brother for their continuous support and encouragement throughout the journey in completing this degree.

My deepest gratitude goes to my supervisor Arild Buland at Equinor ASA, for his advice, assistance and continuous guidance.

Abstract

The use of machine learning algorithms for predictive analytics is making a growing impact in the field of petroleum geosciences. With the increasing cost and time-related factors for obtaining accurate porosity measurements from well logging and coring operations, machine learning (ML) provides a more economical and efficient solution to this challenge.

In this thesis, various ML models are applied to predict porosity in a well penetrating the reservoir interval of the Brent Group to Top Cook formation. The study area is the Statfjord field, located in the Norwegian sector of the North Sea. Statfjord produces oil and associated gas from Jurassic sandstone in the Cook formation, Brent and Statfjord Group.

Sixteen wells with several well logs serve as input features to predict the porosity in a blind well 33/9-4, all located in the field. The machine learning input features are the well logs, feature engineered logs, location points and the measured depth. The logs include: caliper, resistivity, gamma-ray, sonic, density; the engineered logs include: acoustic impedance and facies; the location: x,y,z; and the well's measured depth. The input features are varied and ingested into the ML models to estimate the porosity in the predefined reservoir interval.

The predicted porosity results for the blind well indicated an excellent performance demonstrated by the Bayesian ridge regression, linear regression and random forest models compared to the other ML models used in this study. These three algorithms are highly effective and accurate in predicting porosity with the limited range of the dataset and the results show they can be applied as a more general porosity estimation technique by varying the scale of the data samples and the number of wells.

Table of Contents

List of Tables

List of Figures

1 Introduction

The process of performing a good reservoir characterization and formation evaluation is a critical stage in oil and gas exploration. Porosity is a key characteristic of hydrocarbon-bearing formations. As well-logging and coring operations are time-consuming and expensive to carry out, all wells in typical oil and gas fields are often logged using different tools to measure various petrophysical parameters such as porosity. Furthermore, well logs related to porosity (such as bulk density and neutron porosity) and well logs related to both permeability (such as sonic and nuclear magnetic resonance) often fail to provide satisfactory interpretation results due to empirical parameter uncertainty and response equation adaptability (Zhong et al., 2019). Therefore, it is necessary to establish a low-cost, time-saving, and reliable evaluation method for porosity estimation.

In geosciences, amongst many other machine learning applications, machines have learned to perform rapid reservoir characterizations. Machine learning (ML) has made this process much easier, faster, and economical by learning through uncounted experiences from already explored and developed reservoirs, their rock properties, and the cross-ponding fluid flow behaviour under different circumstances and hence, predicts accordingly. Machine learning is at the forefront of artificial intelligence (AI) technology: a group of data analysis algorithms that include classification, regression, and clustering (Hall, 2016). The ML technique is broadly divided into a supervised and unsupervised group. For supervised ML, the essential members are input features and target output. In this study, multiple AI and ML techniques are compared and discussed in detail to predict porosity using supervised ML algorithms and an advanced deep learning neural network from a series of input features.

Recent advances in machine learning have improved over the years, leading to many of its applications in geoscience. It has been shown that artificial neural networks

(ANNs), as a method of artificial intelligence, can increase the ability of problem-solving to geoscience and petroleum industry problems, particularly in case of limited availability or lack of input data (Ashena et al, 2015).

1.1 Previous ML Applications in Reservoir Characterization

ML methodologies in reservoir characterization has been on steady growth over the years. ML was applied by Al Khalifa et al. (1995) in the prediction of permeability and diagenesis in tight carbonates using various techniques in which the ANN technique provided the best overall prediction method, quantified by the lowest root-mean-square error (RMSE) and highest coefficient of determination value (R^2).

Al-Anazi and Gates (2015) used support vector machine (SVM) to predict Poisson's ratio and Young modulus of reservoir rocks, in which the learning and predictive capabilities of the SVM method were compared to that of a backpropagation neural network (BPNN). The results demonstrate that SVM has similar or superior learning and prediction capabilities to that of the BPNN.

Another case study for the Appalachian Basin in the USA indicated that accurate prediction of facies and fractures in sedimentary rocks could be performed using Bayesian network and Random Forest methods based on petrophysical logs (Bhattacharya & Mishra, 2018).

Active Learning Method (ALM) was used to estimate missing logs in hydrocarbon reservoirs by Bahrpeyma et al. (2015). The regression and normalized mean squared

error (MSE) for estimating density log using ALM were ~ 0.9 and 0.042, respectively. The results, including errors and regression coefficients, proved that ALM was successful in estimating the density.

Another study further investigated the application of SVM in lithology classification, with an observation that SVM performs poor classification results in crystalline rocks when the training samples are imbalanced (Deng et al., 2017). For lithofacies classification, Dell'Aversana (2019) compared six different machine learning methods. The Random Forest and Adaptive Boosting were regarded as slightly more reliable than Naïve Bayes, Decision Tree, and CN2 Rule Induction in lithofacies classification problems, with SVM having a good classification performance.

Yasin et al. (2020) used ML to predict the porosity of the clastic depositional system of the Indus Basin, Pakistan. Their paper presented an approach of joint inversion that combines SVM and particle swarm optimization (PSO) algorithms to predict the porosity's spatial distribution using well logs and seismic data. The results showed that their joint inversion technique led to the most stable prediction of AI and porosity distribution in the lower Goru reservoir of Pakistan's Sawan Field. The tuning of the individual spatial distribution of lithology and porosity from well logs using Gaussian simulation and post-stack seismic inversion using SVM and PSO; revealed favourable matching with the spatial pattern of low AI corresponding with the high porosity and sandstone lithofacies.

1.2 Aim of the Study

This study aims to develop several machine learning models to predict porosity by inputting a series of petrophysical logging data and engineered features to estimate the porosity log values in the oil-bearing reservoir intervals of the Brent Group and Top Cook in a single well.

1.3 Objectives

- Features generation from the following;
    - Location data (x,y,z) and measured depth (MD) of the wells
    - Density (RHOB), sonic (DT), caliper (CALI) and resistivity (RT) logs.
    - Acoustic impedance logs (AI); computed from the product of sonic (DT) and density (RHOB) logs.
    - Facies log; generated from gamma-ray (GR) cut-offs.
- Build training and test models by applying several machine learning algorithms: Bayesian ridge regression, random forest, support vector machine, linear regression, k-nearest neighbour, decision tree and an artificial neural network.
- Predict porosity log values of the defined target zone in a single-blind well (33/9-4).
- Compare the various input features to observe their influence on the predicted porosity results, determine the best machine learning algorithm for this study case, and rank them accordingly.

2 Geological Setting

This chapter provides a brief overview of the structural geology and stratigraphic evolution of the Statfjord field, including a description of the reservoir properties of the field.

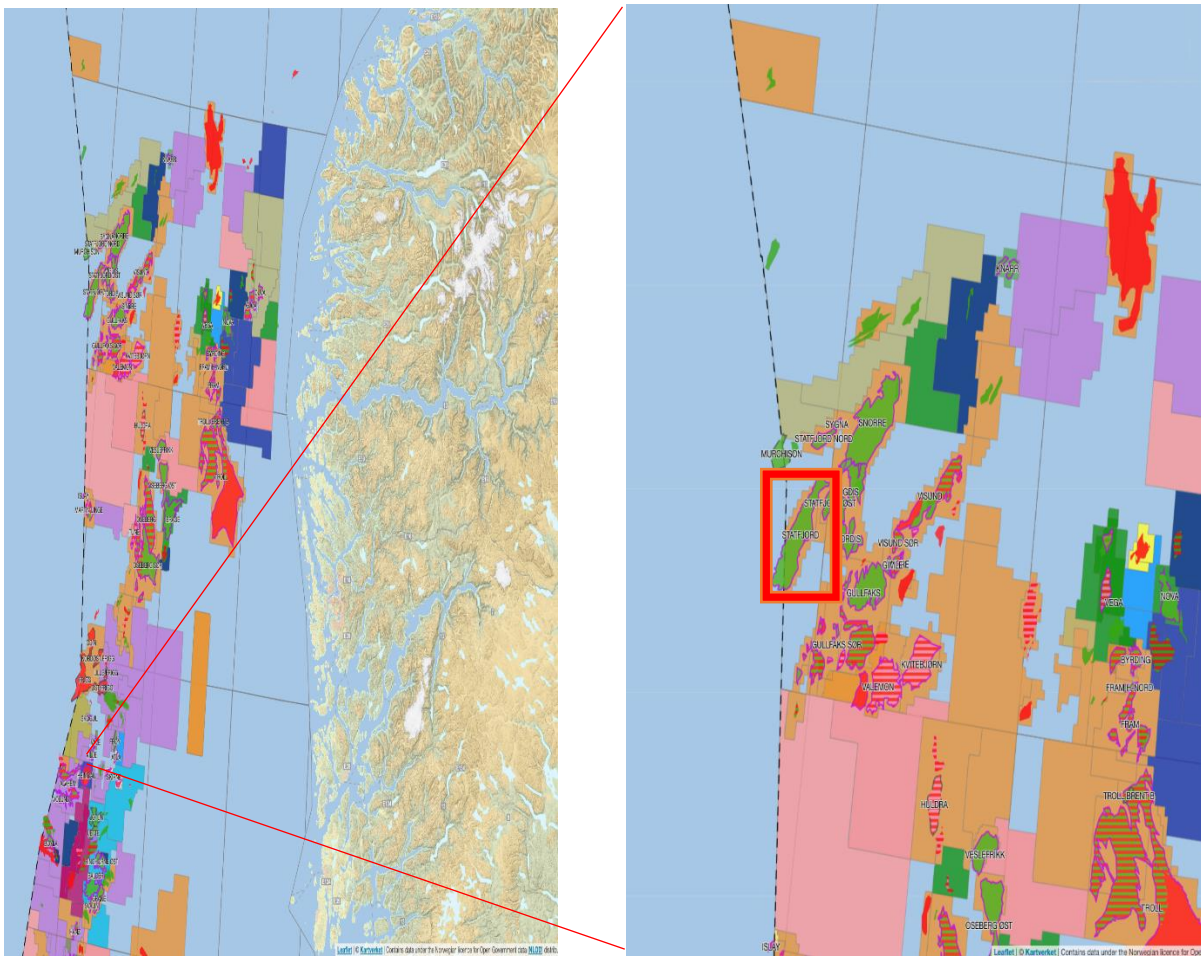


Figure 2.1. Location of the study area in Statfjord field: highlighted in a red box (NPD, 2021)

2.1 Structural and Stratigraphy setting of the Statfjord field

The Statfjord field is the largest oil field in the Northern North Sea. It straddles the Norway/UK boundary and is considered the major field that led to the rise of Norway as a dominant oil-producing nation. Statfjord is located at the southwestern part of the Tampen Spur within the East Shetland Basin. The Tampen Spur is an area where Jurassic-Triassic rocks are structurally high. The major fault trends have a north-south to southeast orientation. Tampen spur is bounded to the east by the West Viking trough marginal fault, which shows a displacement of up to 1500m at the Base Cretaceous level (Spencer et al., 1987). The accumulated hydrocarbon is trapped inside a 6-8° west-northwest dipping rotated fault block comprised of Jurassic-Triassic strata sealed by Middle to Upper Jurassic and Cretaceous shales. Structurally, the field is subdivided into the main field area characterized by relatively undeformed west to northwest dipping strata and a heavily deformed east flank area characterized by several phases of 'eastward' gravitational collapse (Gibbons et al. 2003).

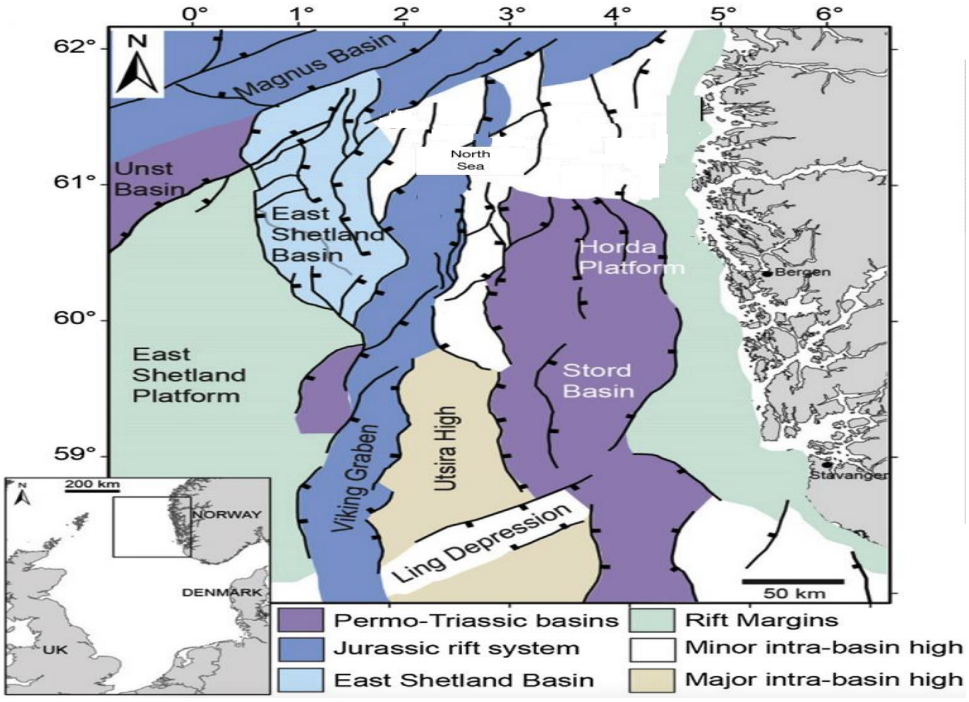


Figure 2.2. A simplified map showing the main structural elements of the Northern North Sea (modified from Duffy, 2015)

The NW-SE extension was initiated during the Late Triassic times (Figure 2.1.1); this is expressed by numerous listric faults which are not connected to the underlying Permian-early Triassic age horst and graben features. Extention continued throughout Jurassic time, culminating during late Middle Jurassic to Early Cretaceous times with domal uplift and erosion (Ziegler, 1981)

A regional unconformity is widely developed within the Lower and Middle Jurassic succession. This unconformity was termed the Mid-Cimmerian unconformity in earlier literature (Ziegler 1990a) and identified as the 'Intra-Aalenian Unconformity' by Underhill and Partington (1993). (Evans et al. 2003).

However, this unconformity is shown to cover a wider length over most of the study area as this stratigraphic gap widens as it extends eastwards from the West Province and further southwards in the North-Central Province. In significant areas of the Central Province and large parts of the East Province, the Lower Jurassic strata are noticeably absent, and Middle Jurassic strata are observed to rest unconformably on Triassic or older rocks.

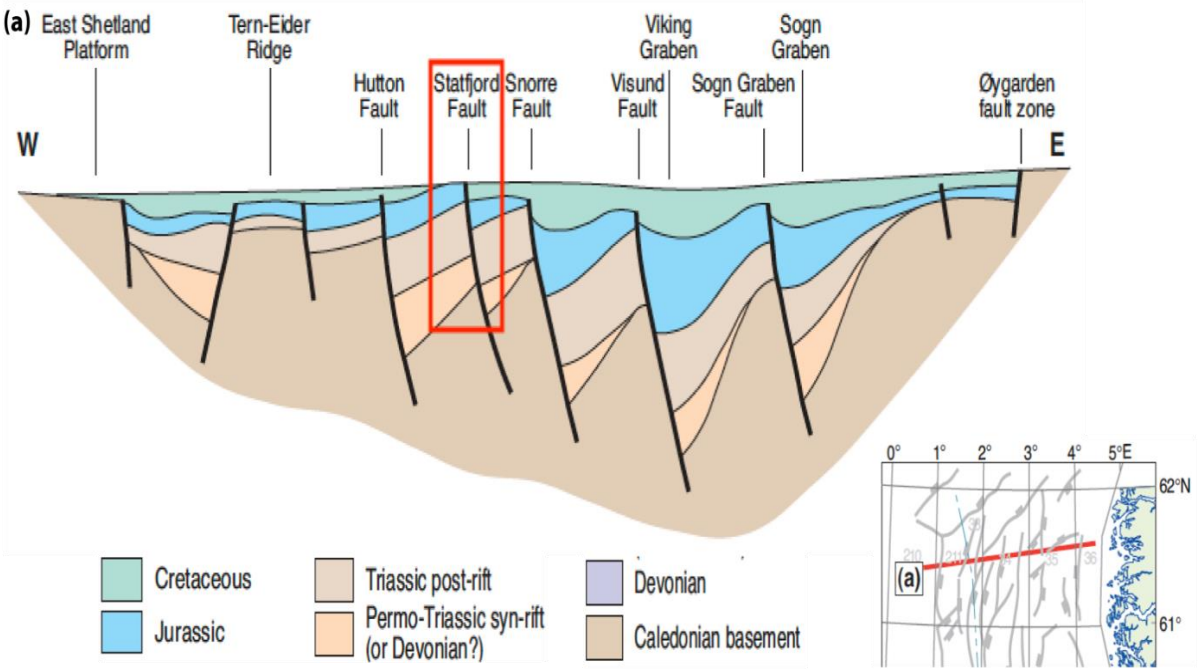


Figure 2.3. Diagrammatic profiles across the Northern North Sea, modified from (Evans et al. 2003).

Figure 2.3 shows the diagrammatic profiles across the Northern North Sea. They indicate that the position of the Viking Graben moves westward across the northern North Sea from north to south. The field's reserves are located in three distinct reservoirs: Middle Jurassic; which holds the Brent Group's deltaic sediments, Lower Jurassic; which consists of marine-shelf sandstones and siltstone sediments of the Dunlin Group and the Upper Triassic-lowermost Jurassic fluviatile sediments of the Statfjord Formation. The Brent Group lithostratigraphy is essentially simple, consisting of five formations, which from the base upwards are: Broom, Rannoch, Etive, Ness, and Tarbert (Bowen 1975; Deegan & Scull 1977; Vollset & Dore 1984). The first formal lithostratigraphic nomenclature for this offshore area was proposed in a UK–Norwegian collaborative report by Deegan and Scull (1977) that spanned these two national sectors.

2.2 **Reservoir Properties of the Statfjord field**

The main oil-producing reservoirs of the field are sandstones of the Middle Jurassic Brent group and the Lower Jurassic/Upper Triassic Statfjord formation. The reservoirs dip to the west, and the field is bounded on the east by a boundary fault system. These accumulations are sealed by Upper Jurassic and Cretaceous shales (Aadland et al. 1994) as seen in Figure 2.3 which shows a representational west-east cross-section through the field. The majority of the reserves within the Brent Group and Statfjord formation sediments exhibit good to excellent reservoir properties with porosities ranging from 20-30%, permeabilities going up to several darcies and an average net-to-gross of 60-75% (Gibbons et al., 2003).

Spencer et al. (1987) described the Statfjord formation as 200m thick in the Statfjord field area but thins progressively to the northeast, with thickness variations locally controlled by early Cimmerian block rotation transgressed in the course of Early Jurassic times.

The sandstones and siltstones of the Dunlin Group have more inferior reservoir properties where the best reservoir unit exhibits an average porosity of 22%, an average permeability of 300 raD, and net-to-gross of 45% (Gibbons et al. 2003).

The Upper Brent reservoir (Figure 2.4) consists of the Tarbert and Ness formations. The Tarbet has good horizontal permeability ranging from 2 to 3 darcies, as excellent vertical communication exists. Although restricted communication occurs between the single sand bodies in the fluvial Ness formation, permeability is in the 1-darcy range in each sand interval. The Lower Brent consists of Etive, Rannoch, and Broom formations. The Etive generally holds very clean sands with excellent reservoir properties and permeability in the 5-6 darcy range. Initial oil saturation is >90%. Permeability in the Rannoch Formation ranges from poor (10 mD) to moderate at the base to good (1 darcy) at the top. (Aadland et al. 1994).

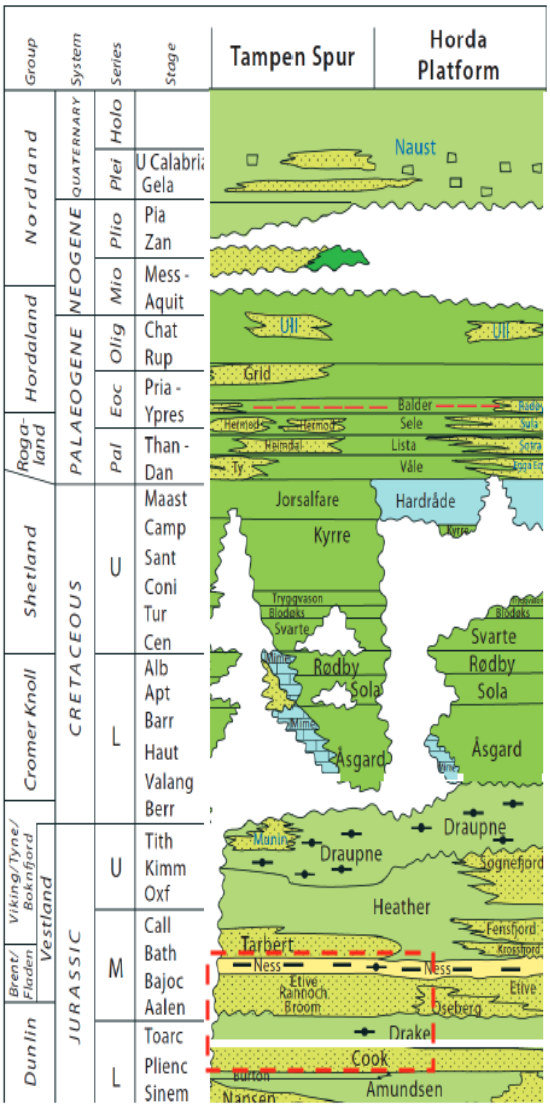


Figure 2.4. Lithostratigraphic chart of the North Sea with its main sub-sections. (NPD, 2020). Red box outlines the key lithostratigraphic intervals that have been studied in this thesis

The most important source rock for the Statfjord field petroleum system lies in the Upper Jurassic – The Kimmeridge clay (UK) and the Draupne formation(Norway). The significance of the source rock is shown by its estimated potential of 80 litres of oil being generated per cubic meter of rock (Spencer et al. 1987)

3 Machine Learning

Machine learning is a branch of artificial intelligence (AI) and is defined as a training process of giving computers the capability to learn and act using sets of algorithms, and there are various definitions of machine learning from different perspectives. In Nikhil's (2017) work, he describes machine learning as: instead of teaching a computer a massive list of rules to solve the problem, give it a model with which it can evaluate examples and a small set of instructions to modify the model when it makes a mistake.

Alpaydin (2014) describes machine learning as applying statistics principles in building mathematical models because the core task is to make inferences from a sample. He further describes it as programming computers to optimize a performance criterion using example data or past experience. When a model is defined up to some parameters, learning is executing a computer program to optimize the model's parameters using the training data or past experience. This model could be predictive to make predictions in the future, or descriptive to gain knowledge from data, or possibly both.

In ML, uncertainty arises in many forms: what is the best prediction about the future given previous information? What is the best ML model to explain the dataset? What are the successive measurements to be performed?. The probabilistic approach to machine learning is strongly related to the field of statistics but slightly differs in its emphasis and terminology (Murphy, 2012).

Machine learning is typically divided into two main types; the supervised or predictive learning approach and the unsupervised or descriptive learning approach. The third type of machine learning, known as reinforcement learning, is somewhat less commonly used.

3.1 **Artificial Neural Networks**

Artificial Neural Networks (ANN) are computing systems modeled on the biological brain structure, and can be used for ML and AI. It is made up of an abstracted model of connected neurons whose unique link and arrangement solve computerized application problems in various fields such as statistics, technology, or economics.

Unsupervised learning in an ANN attempts to get the ANN to "understand" the structure of the provided input data "on its own". The biological neuron is simulated in an ANN by an activation function. In classification tasks (e.g., identify spam e-mails), the activation function has a "switch-on" characteristic – which means, once the input is greater than a specific value, the output should change state, e.g., from 0 to 1, from -1 to 1 or from 0 to > 0. This simulates the "turning on" of a biological neuron. A very common activation function that is used in ANN is the sigmoid function.

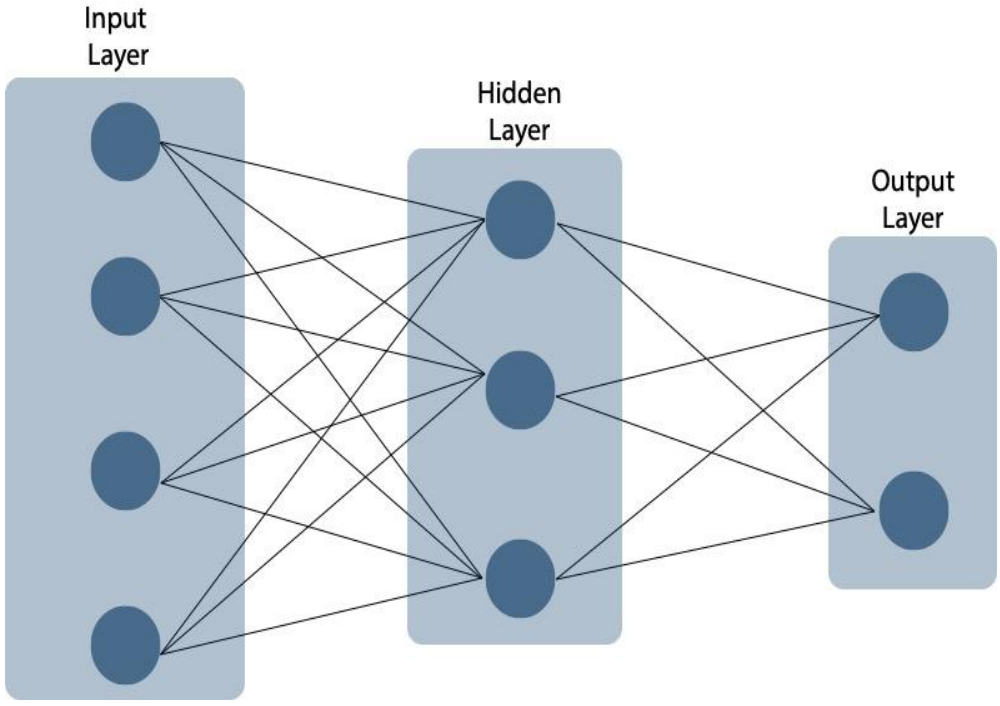


Figure 3.1. Artificial Neural Network (Feed-forward)

Neural networks learn in two steps, feedforward as in Figure 3.1 (using the activation function) and back propagation, which is broken down into two stages: computing for the cost and minimizing the cost.

The cost is the difference between the predicted value from the network and the expected value from the dataset. The larger the cost the more significant the error, with the objective of having the smallest possible cost. To achieve this, minimizing the cost through altering the weights and biases is the primary goal.

As in Figure 3.1, the input layer is often illustrated with one node for each feature of the p-dimensional vector x and simplified to one node for x in a graph. The number of hidden nodes are flexible but usually less than the number of input nodes. For many applications, there is only one output node such as real number y for regression and categorical variable c for classification (Buland, 2020)

3.2 K Nearest Neighbour

Cover and Hart (1967) were pioneers of the theoretical analysis of nearest neighbours, covering both regression and classification as exceptional prediction cases in general. The principle behind nearest neighbour technique is to find a predefined number of training samples closest in the distance to the new point and predict the label from these. The number of samples can be a user-defined constant (k-nearest neighbour learning) or vary based on the local density of points (radius-based neighbour learning).

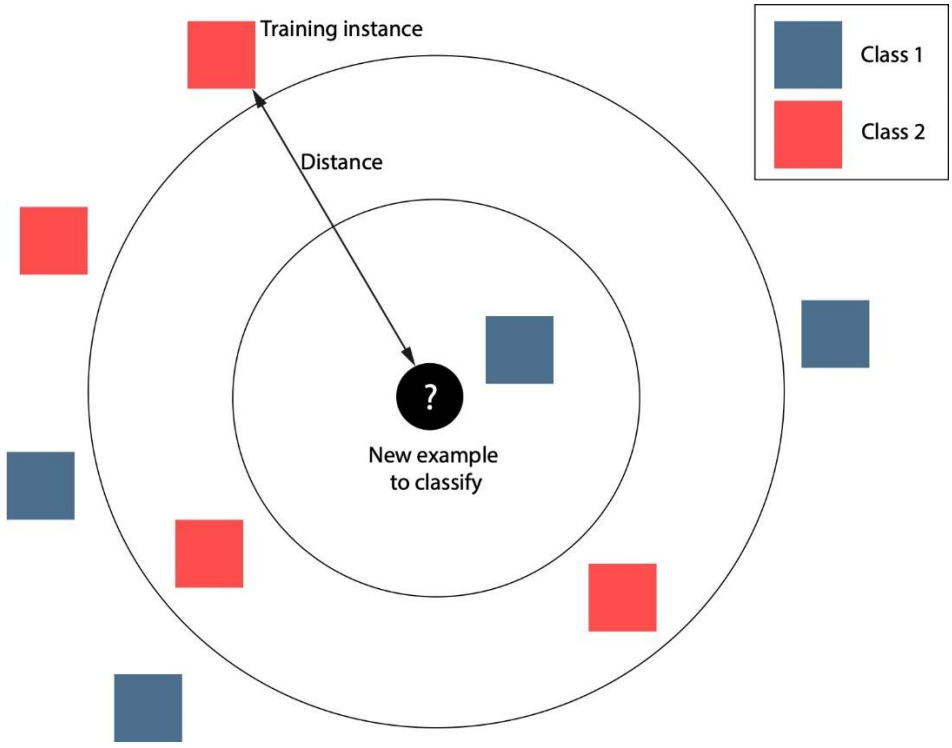


Figure 3.2. Example of KNN classification. (modified from Bronshtein, 2017)

A typical K nearest neighbour (KNN) classification, as in Figure 3.2, shows a test sample (inside the circle), should be classified either to the first class of blue squares

or to the second class of red squares. If k = 3 (outside circle), it is assigned to the second class because there are two red squares and only one blue square inside the inner circle. If, for example, k = 5, it is assigned to the first class (three blue squares vs. two red squares outside the outer circle).

KNN algorithm is based on feature similarity: How closely out-of-sample features resemble our training set determines how we classify a given data point (Bronshtein, 2017). The distance can generally be any metric measure, with the standard Euclidean distance the most common choice. Neighbours based methods are non-generalizing machine learning methods since they recollect all of its training data (possibly transformed into a fast indexing structure). This algorithm acts as a uniform interface to three different nearest neighbours algorithms.

For regression, assume the nearest neighbour of a vector x is the x_i closest to it. The k nearest neighbours are the k vectors x closest to x_i (whether or not x is also one of the x_i). We often need a way of keeping track of the indices of the neighbours, so $NN(x, j)$ is written for the index of j^{th} the nearest neighbour of x.

The KNN estimate of the regression function is then the average value of the response over the KNN:

$$\mu(x) = \frac{1}{k} \sum_{j=1}^{k} y_{NN}\,(x, j) \tag{1}$$

And then threshold it:

$$\mathrm{c}(x) = 1(p(x) \geq 0.5) \tag{2}$$

Where;

p is the probability distribution function,

c is the class.

3.3 **Random Forest**

In Random Forest (RF) classification, multiple trees are used to train and predict samples. RF algorithm is being used in an increasing number of engineering research studies but is still rarely used for porosity and permeability classification (Sun et al., 2021). Since the random forest contains multiple decision trees, the output category of the classifier is the category with the largest output of all single decision trees.

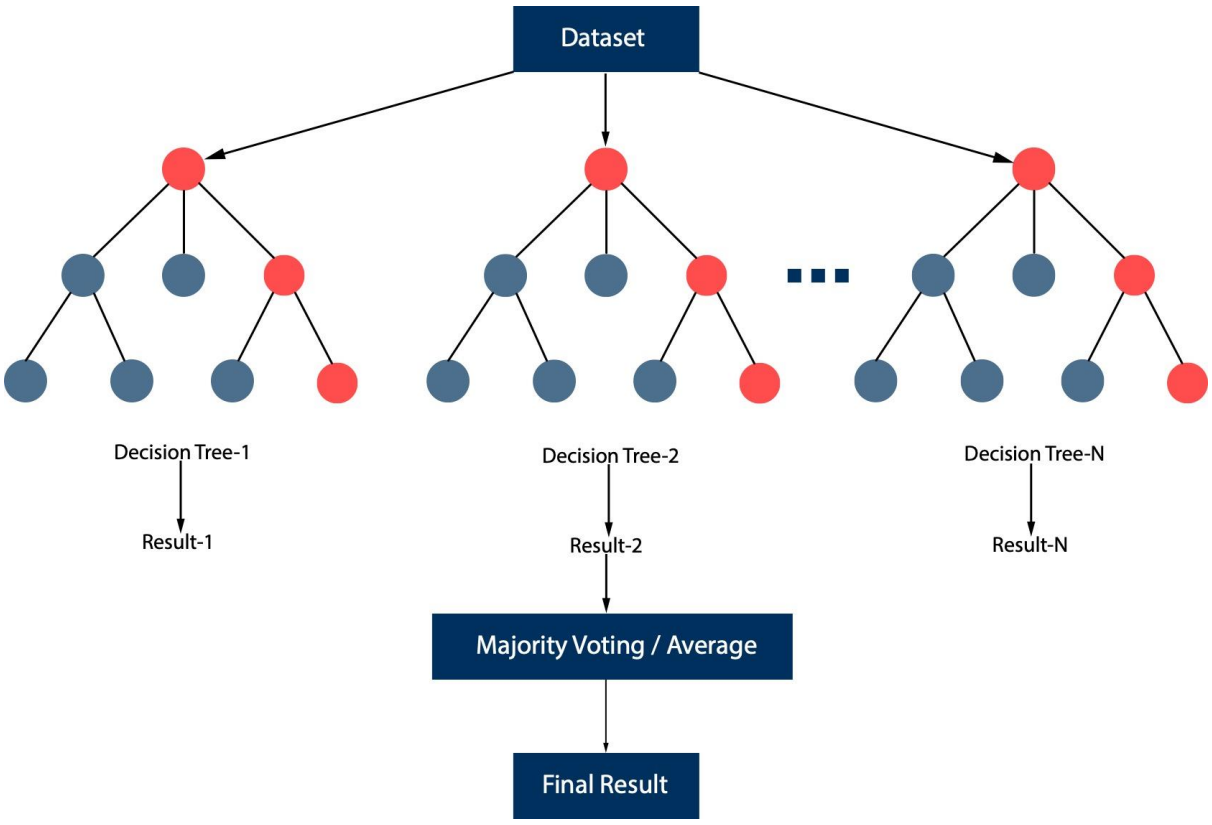


Figure 3.3. Illustration of random forest algorithm structure

As in Figure 3.3, The number of training samples is N, and the number of feature types is M. Inputting m features to determine the decision result of the previous node in the decision tree ($m < M$) using the put-back method to sample N times (i.e., bootstrap sampling) to form a training set, unsampled samples are used to predict and evaluate the error. For each node, m features are randomly selected, and decisions for each

node on the decision tree are determined based on these features. Then, according to m features to calculate the best split mode. One leaf node of the decision tree cannot continue to split, or all samples point to the same category, and each tree will grow ultimately without being pruned.

The random forest algorithm can produce a high-accuracy classifier for classification problems with a large number of features. It can assess the importance of feature parameters when determining categories, and it can balance errors and maintain accuracy for missing data or unbalanced data.

3.4 **Decision Tree**

The decision tree (DT) is a supervised learning algorithm for predictive modeling approaches in statistics, data mining, and machine learning, Figure 3.3. It uses a decision tree (predictive model) to go from observations about an item (branches) to conclusions about the item's target value (leaves). The decision tree used in this study is the regression tree. The regression tree is a recursively constructed binary decision tree based on minimizing the square error (Sun et al., 2021). The regression tree is described below:

Suppose vectors x and y are input variable and output variable, respectively, and y is a continuous variable, given a training data set:

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \quad (3)$$

A regression tree divides the input space (i.e., the feature space) into M units $\{R_1, R_2, \dots, R_m\}$ and each leaf node of the regression tree corresponds to a unit, which

correspondingly has a fixed output value c_m. When the input feature is x, the regression tree will determine it to a leaf node, and the output value c_m corresponding to the leaf node is used as the output of the regression tree. In this way, the regression tree model can be expressed as:

$$T(x) = \sum_{m=1}^{m} c_m I(x \in R_m) \tag{4}$$

Where:

$I(x \in R_m)$ is the index function when the regression tree determines that x belongs to R_m its value is 1; otherwise, it is 0.

The goal of establishing a regression tree is to minimize the square error for data set D and choose the appropriate spatial partitioning method (i.e., the way the decision tree is generated) and the corresponding output values.

$$\sum_{x_i \in D} (y_i - T(x_i))^2 \tag{5}$$

Firstly, the appropriate space division method should be chosen. At each decision node, the j^{th} dimension of the variable x and the corresponding threshold s are selected as the segmentation feature and the segmentation threshold, and the node divides the space into two regions:

$$R_1(j, s) = \{x | x[j] \leq s\} \tag{6}$$

And the second region as:

$$R_2(J, s) = \{x|x[j] > s\} \tag{7}$$

The optimal segmentation feature j and the segmentation threshold s at the node are given by:

$$min_{j,s}\left[min_{c_1}\sum_{x_i \in R_1(j,s)}(y_i - c_i)^2 + min_{c_2}\sum_{x_1 \in R_{2(j,s)}}(y_{i-}{}_{c_i})^2\right] \tag{8}$$

The node divides the sample set into two sub-sample sets according to the segmentation feature j and the segmentation threshold s. The specific purpose of Eq. (8) is that the variance of the two sub-sample sets is as tiny as possible. The values of c_1 and c_2 in Eq. (8) are the mean values of the samples, and the choices of j and ss are determined by traversal. The segmentation threshold s is a continuous variable, but its value can be selected according to the actual distribution of the sample to select a suitable specific value without continuous traversal.

Secondly, the output value of the regression tree should be determined. For each sub-region R_m (the leaf node of the tree), the corresponding output value c_m can directly select the category mean values, i.e.,

$$c_m = \arg min_c \sum_{x_i \in R_m}(y_i - c)^2 = average(y_i| \in R_m) \tag{9}$$

3.5 **Support Vector Machine**

Support vectors are power-supervised training machine learning methods for segmentation. Vapnik et al. (1995) first proposed SVMs as one effective algorithm for model pattern recognition. It is a fundamental method that the SVMs can solve nonlinear functions by leveling the data into a higher-dimensional space and

introducing an optimal hyperspace in the space through kernel functions. SVMs can be further divided into support vector classification (SVC) and support vector regression (SVR). SVR is developed on the basics of SVC with the same methodology. Multiple types of kernels have been developed to map data into differing dimensions. If the kernel transformation function does not fully separate our data, a slack error variable is used to create a soft margin decision function for data separation (Boyle, 2011). Figure 3.4 illustrates an example SVM decision function and displays the margin. Therefore, some definitions and properties of SVC are restated as follows: SVM is a discriminative classifier designed to separate by a hyperplane. The hyperplane is used to divide the margins as wide as possible between the points of different categories (Qiang et al., 2020). These are evaluated according the subset of training sample that lie closest to the boundary and called support vectors (Burges, 1998), as illustrated in Figure 3.5.

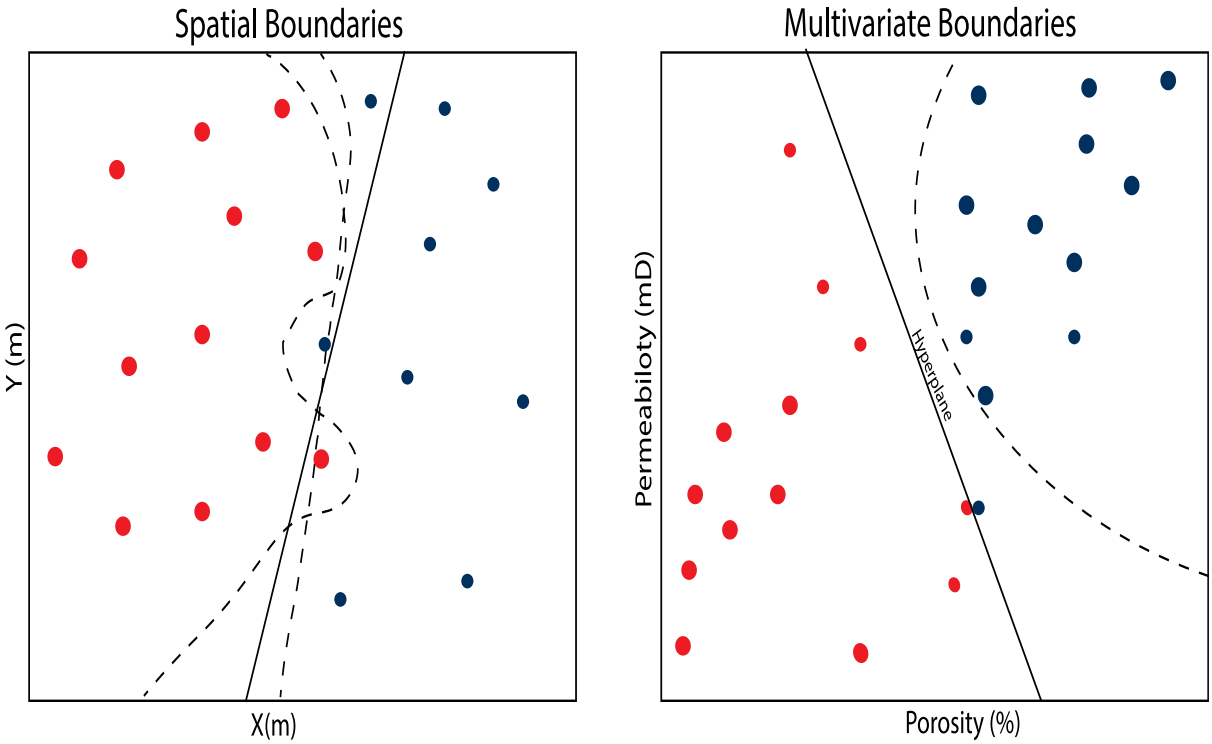


Figure 3.4. Illustration of boundary decision: spatial (left) and multivariate (right), (modified from Pyrcz, 2020).

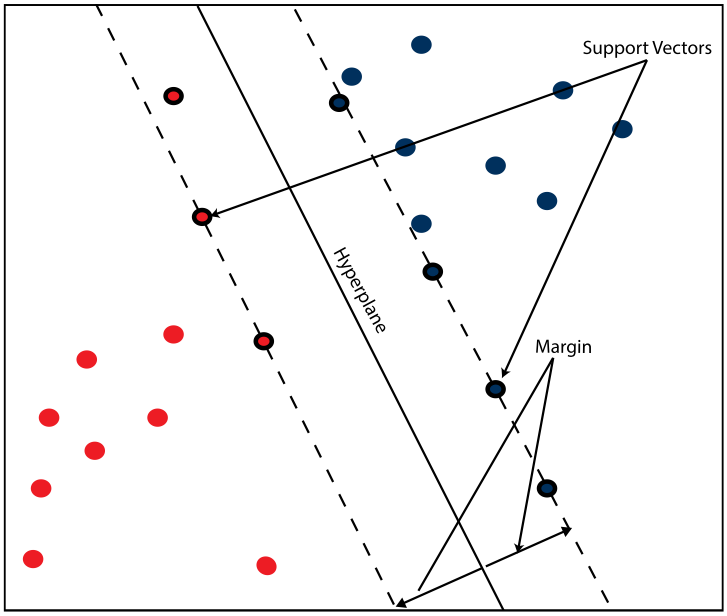


Figure 3.5. Support vector illustration

The support vector model can further be explained mathematically by;

Solve for the hyperplane:

$$f(x) = x^T\beta + \beta_0 \tag{10}$$

$f(x)$ is proportional to the sine distance from the boundary, - one side and + the other, 0 on the boundary

$$G(x) = sign(f(x)) \tag{11}$$

Where x is a vector $x_j, j = 1, \ldots, m$ predictor features.

We can represent the constraint: where all data of each category must be on its correct side of the boundary by:

$$y_i\,(x_I^T\beta + \beta_0) \geq 0 + \tag{12}$$

If y_i is the response feature with categories -1 or +1

When the training data categories overlapping it would not be possible, not desirable, to develop a decision boundary that perfectly separates the categories which this condition would hold. This makes it needful to have a model that allows for some misclassification:

$$y_i\,(x_I^T\beta + \beta_0) \geq M - \xi_i \tag{13}$$

Where M is inside the boundary and ξ_i the is error relative to M.

Solving for the SVMs using quadratic with linear inequality constraint Eq. (13). We express the previous relationship using convex optimization problem (to avoid local minimum issues) as

$$min_{\beta,\beta_0}\left(\frac{1}{2M^2} + \mathrm{C}\sum_{\mathrm{I}=1}^{\mathrm{N}}\xi_{\mathrm{I}}\right) \tag{14}$$

Subject to $\xi_i \geq 0, \quad y_i\,(x_I^T\beta + \beta_0) \geq M - \xi_i$ (15)

As in Eq. (15), the parameters of the plane are found to maximize the margin while minimizing the error. The C, which is constant and the hyperparameter is included to weigh the sum of errors ξ_i, higher C will result in reduced margin and lead to overfitting.

3.6 **Linear Regression**

Regression analysis is a statistical technique for investigating and modeling the relationship between variables. (Montgomery et al., 2012). Linear Regression can be defined as a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. It can be used to predict values within a continuous range rather than trying to classify them. It is common to discuss the complexity of a regression model like linear regression, which refers to the number of coefficients used.

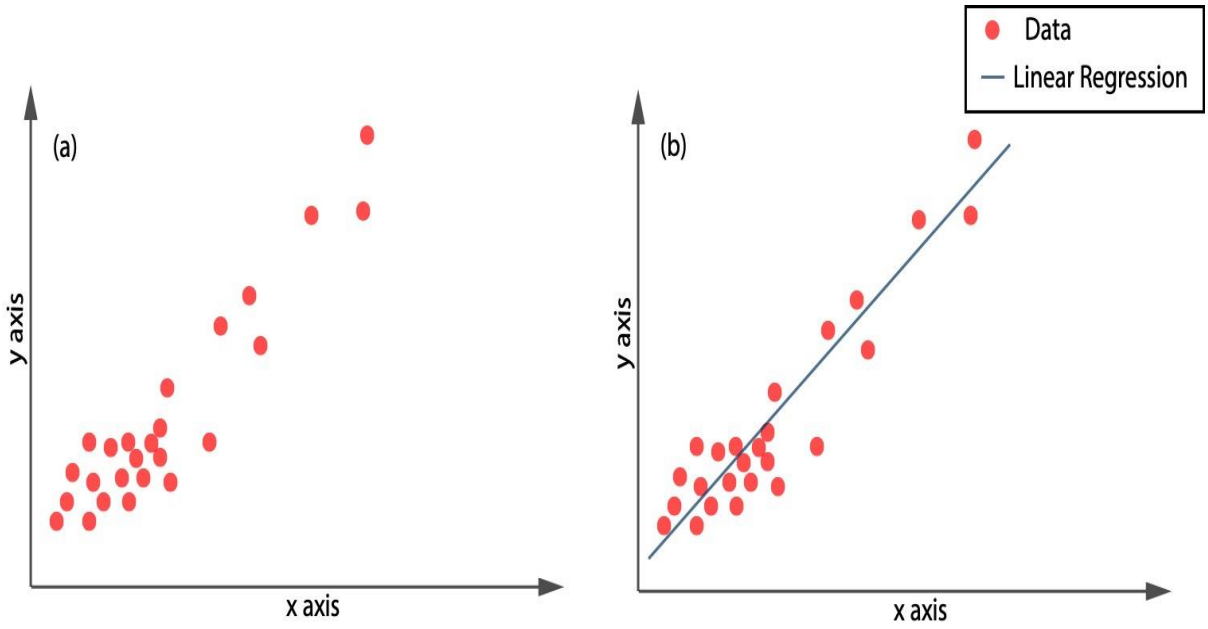


Figure 3.6. (a) Scatter diagram for x and y (b) Straight-line relationship between x and y

Figure 3.6a displays a relationship between data points x and y. The idea is that the data points typically, but not precisely, fall along a straight line. Figure 3.6b illustrates this straight-line relationship.

The x signifies data from the x-axis, and y signifies the y-axis. A straight line relating these two variables can be written as:

$$y = \beta_0 + \beta_1 x \tag{16}$$

Where; β_0 is the intercept and β_1 the slope.

When the coefficient becomes zero, it effectively removes the influence of the input variable on the model and, therefore, from the model's prediction.

As the data points do not fall precisely on a straight line, Eq. (16) is modified to account for this. Let the difference between the observed value of y and the straight line $(\beta_0 + \beta_1 x)$ be a statistical error ε. The error is a random variable that accounts for the failure of the model to fit the data exactly and the linear regression model is given by

$$y = \beta_0 + \beta_1 x + \varepsilon \tag{17}$$

In general, the response variable y may be related to k regressors, $x_1, x_2, \ldots, x_k$ so that

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \beta_k x_k + \varepsilon \tag{18}$$

Eq. (18) is the multiple linear regression model as more than one regressor is involved. The adjective linear is employed to indicate that the model is linear in the parameters $\beta_0, \beta_1, \ldots, \beta_k$, not because y is a linear function of the $x's$.

However, a regression model does not imply a cause-and-effect relationship between the variables. Even though a solid empirical relationship may exist between two or more variables, this cannot be considered evidence that the regressor variables and the response are related in a cause-and-effect manner (Montgomery, 2012).

3.7 **Bayesian Ridge Regression**

Bayesian ridge regression is a regression model defined in probabilistic terms, with explicit priors on the parameters. The choice of priors can have a regularizing effect. The end product of a Bayesian Regression model is obtained from a probability distribution compared to traditional regression techniques, where the output is obtained from a single value of each attribute. However, Bayesian ridge regression is used relatively rarely in practice (Pasanen, 2015).

The output, 'y' in Eq. (16), is generated from a normal distribution (where mean and variance are normalized). Bayesian regression does not aim to find the model parameters but the model parameter's posterior distribution, not just the output y, but the model parameters are also assumed to come from a distribution. The expression for Posterior is :

$$Posterior = \frac{Likelihood * Prior}{Normalization} \tag{19}$$

- Posterior**:** The probability of an event to occur; say, H, given another event; say, E has already occurred. i.e., P(H ∣ E).
- Prior: The probability of an event H has occurred prior to another event. i.e., P(H)
- Likelihood: A likelihood function in which some parameter variable is marginalized.

Eq. (19) a simple expression of Bayes theorem, the fundamental underpinning of bayesian inference, which is:

$$P(\beta|y,X) = \frac{P(y,X|\beta)P(\beta)}{P(y,X)} \tag{20}$$

Where;

$P(\beta|y,X)$ is the model parameter's posterior probability distribution given the inputs and outputs. This is equal to $P(y|\beta,X)$ representing the likelihood of the data multiplied by the prior probability of the parameters and derived by a normalization constant.

From Eq. (20), in contrast to ordinary least square (OLS), we have a posterior distribution for the model parameters, proportional to the likelihood of the data multiplied by the prior probability of the parameters. As the number of data points increases, the value of likelihood will increase and become much larger than the prior value. In an infinite number of data points, the values for the parameters converge to the values obtained from OLS.

Additionally, to begin the regression process with an initial estimate (the prior value) and as more data points are covered, the model is more precise**.** For Bayesian ridge regression, a large number of training data is needed to make the model accurate.

Consider a linear model; if 'y' is the predicted value, then:

$$\hat{y}(w,x) = w_0 + w_1 x_1 w_p x_p \tag{21}$$

Where;

'w' is the vector w. w consists of $w_0, w_1, \ldots$'x' is the value of the weights:

$$w = (w_1, \ldots w_p) \tag{22}$$

For the Bayesian regression to attain a fully probabilistic model, the output 'y' is assumed to be the Gaussian distribution around X_w as shown below:

$$p(y|X,w,\alpha) = N(y|X_w,\alpha) \tag{23}$$

Where; α is a hyper-parameter for the prior gamma distribution, this leads it to be assessed as a random variable.

The Bayesian ridge regression is given as a mathematical expression:

$$p(w|\lambda) = N(w|0, \lambda^{-1}) \tag{24}$$

Where; α is the shape parameter for the gamma prior to the α parameter and λ is the shape parameter for the gamma prior to the λ parameter. A linear regression model is formulated, considering probability distributions before seeing the data instead of just training data.

4 Data

Equinor ASA provided the data used for this study. The original data obtained comprised of well log data from 610 wells with well tops.

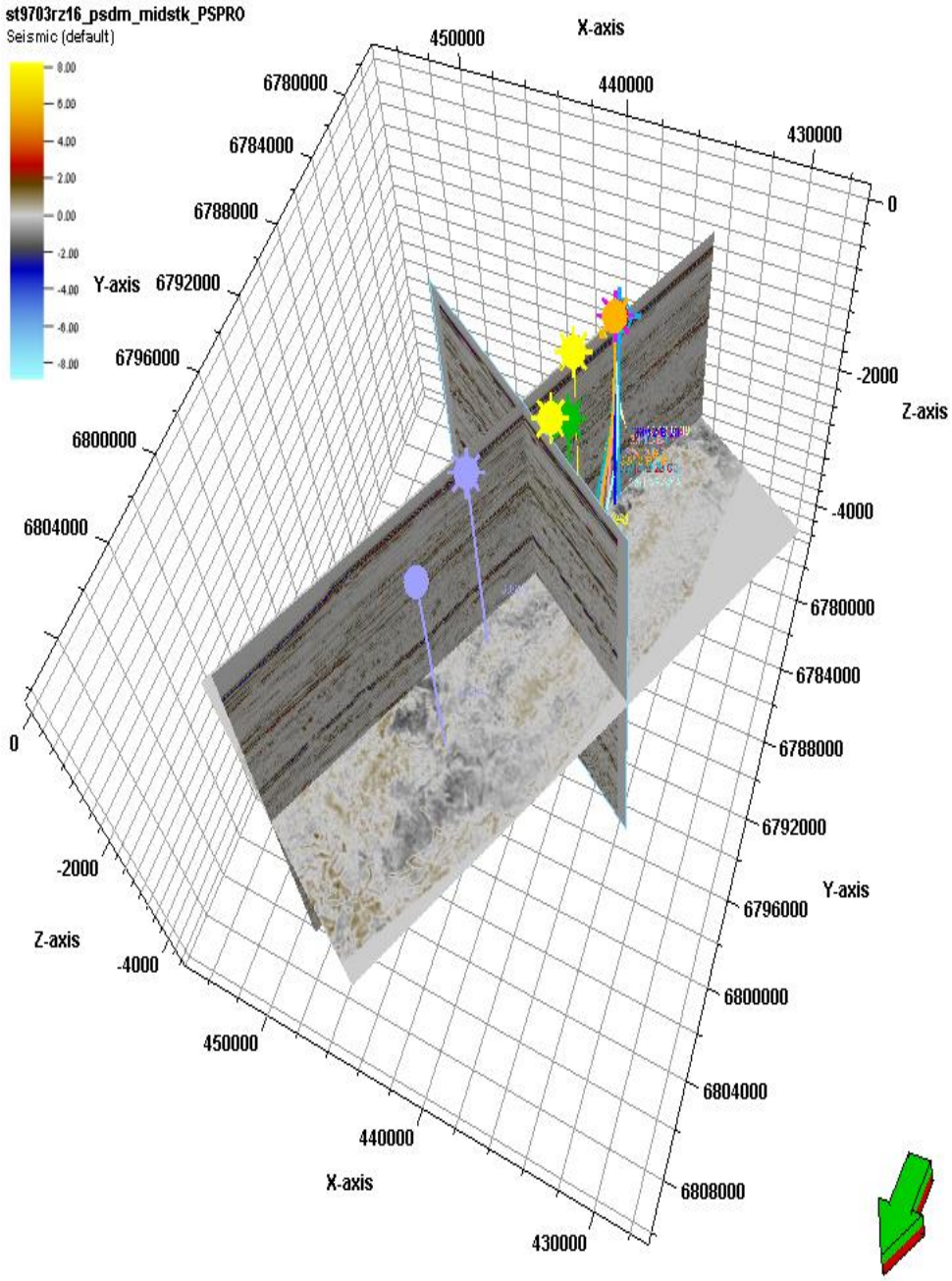


Figure 4.1. The dataset containing the seismic cube and wells

Data from seventeen (17) spatially distributed wells are used for this study, as shown in Table 1. The wells encompass the study area, and several logs are available, providing crucial knowledge of target reservoir interval. In addition, checkshot surveys are available for all the wells. Wireline logs included are the gamma-ray (GR), density (RHOB), sonic (DT), porosity (PHIT), caliper (CALI), and resistivity (RT).

Table 1. Well dataset and corresponding well logs

Wells	GR (gAPI)	DT (s/ft)	RHOB (g/cm3)	PHIT (m3/m3)	CALI (in)	RT (ohm)
33/12-1	✔	✔	✔	✔	✔	✔
33/12-2	✔	✔	✔	✔	✔	✔
33/12-B-17	✔	✔	✔	✔	✔	✔
33/12-B-18	✔	✔	✔	✔	✔	✔
33/12-B-20	✔	✔	✔	✔	✔	✔
33/12-B-28 B	✔	✔	✔	✔	✔	✔
33/12-B-29-T2	✔	✔	✔	✔	✔	✔
33/12-B-3	✔	✔	✔	✔	✔	✔
33/12-B-38	✔	✔	✔	✔	✔	✔
33 /12-B-38 A	✔	✔	✔	✔	✔	✔
33/12-B-39	✔	✔	✔	✔	✔	✔
33/12-B-7	✔	✔	✔	✔	✔	✔
33/12-B-8	✔	✔	✔	✔	✔	✔
33/9-1	✔	✔	✔	✔	✔	✔
33/9-4	✔	✔	✔	✔	✔	✔
33/9-A-41	✔	✔	✔	✔	✔	✔
33/12-B-28 C	✔	✔	✔	✔	✔	✔

5 Methodology

Figure 5.1. General workflow for this study

In this chapter, the workflow and various methodologies applied for the study are described. In addition, the various ML methods predefined in chapter 3 are applied in predicting the porosity value for well 33/9-4 (blind well).

A generic workflow in Figure 5.1 provides an overview of the methods that are applied in the study. Firstly, data exploration was carried out; this involved sorting and filtering the wells to find the wells that had the target reservoir interval well tops. This was followed by computing for the acoustic impedance (AI) log; derived from the product of the sonic and density logs in Petrel. Subsequently, facies classification was done, and then other essential features needed for the ML models extracted. Finally, ML training and test were carried out to predict the assigned label (porosity), using the various input features, and then concluded with a performance evaluation to compare the different ML results.

Two key applications used for this study were Schlumberger's Petrel 2020 software and Anaconda's Jupyter notebook. This notebook uses the Python programming language containing a suite of imported libraries. The Python libraries used for this study include but not limited to; Pandas (data frame manipulation), NumPy (numerical operation),

Seaborn and Plotly (graph and visual plots), Sci-kit learn (machine learning), and Tensorflow/Keras (artificial neural network)

5.1 Data Preparation

Having a good ML model always begins with utilizing a suitable input dataset. Data preparation is the first and most crucial step in building good models. The data preparation process for this thesis involved sorting and filtering the provided dataset and extracting the key wells which had the BCU and Top Cook well tops in them.

Seventeen wells were chosen as they had the reservoir interval needed for this study and were spatially distributed across the field. Furthermore, as there was no horizon for the Top Cook available in the dataset, only the well logs was used in this study.

5.1.1 Well Log Data Analysis and QC

This step involves a QC of the selected wells and their corresponding well logs. Data acquisition comes with its limitations due to numerous pitfalls in methods of acquisition which can reduce overall precision and accuracy, resulting in reduced confidence and robustness. The fact that datasets can often be incomplete or sparse is the primary motivation for most operators to focus on maximizing and fully integrating all information available. Also, washouts and other bad borehole conditions affect the log readings and lead to wrong interpretations. Well tops were compared to available litho-stratigraphic markers from NPD. All well logs were thoroughly reviewed to rigorous QC and condition the data to ensure the quality of the input data moving forward.

	Well identifier	Surface	X	Y	Depth	MD
87	33/12-1	BCU	437055.69	6786258.21	-2384.02	2411.02
90	33/12-1	Cook Fm. SF Top	437055.69	6786258.21	-2540.84	2567.84
95	33/12-2	BCU	438628.87	6788583.73	-2459.61	2484.61
98	33/12-2	Cook Fm. SF Top	438628.87	6788583.73	-2500.15	2525.15
344	33/12-B-17	BCU	437567.06	6786890.76	-2375.57	2522.07
347	33/12-B-17	Cook Fm. SF Top	437576.57	6786899.55	-2446.08	2593.77
368	33/12-B-18	BCU	438700.21	6787748.77	-2472.17	3463.66
373	33/12-B-18	Cook Fm. SF Top	438750.58	6787786.33	-2524.18	3545.25
473	33/12-B-20	BCU	437168.91	6786669.20	-2393.66	2487.33
476	33/12-B-20	Cook Fm. SF Top	437159.22	6786725.66	-2536.22	2640.99
642	33/12-B-28 B	BCU	438336.81	6787779.52	-2422.82	3354.66
645	33/12-B-28 B	Cook Fm. SF Top	438345.44	6787802.89	-2468.84	3407.00
647	33/12-B-28 C	BCU	438700.10	6787992.38	-2467.84	3683.51
650	33/12-B-28 C	Cook Fm. SF Top	438731.75	6788021.11	-2495.55	3734.46
668	33/12-B-29 T2	BCU	439284.20	6786956.84	-2545.85	3784.58
672	33/12-B-29 T2	Cook Fm. SF Top	439452.57	6786971.84	-2636.15	3976.30
675	33/12-B-3	BCU	437564.01	6788399.56	-2365.68	3272.23
677	33/12-B-3	Cook Fm. SF Top	437597.29	6788628.16	-2584.10	3590.30
812	33/12-B-38	BCU	436628.19	6784320.55	-2513.16	3643.86
815	33/12-B-38	Cook Fm. SF Top	436615.08	6784266.70	-2571.95	3724.68
820	33/12-B-38 A	BCU	437614.78	6787179.00	-2370.04	2921.51
823	33/12-B-38 A	Cook Fm. SF Top	437673.68	6787279.03	-2454.91	3065.36
832	33/12-B-39	BCU	437272.07	6784945.28	-2450.66	3117.83
835	33/12-B-39	Cook Fm. SF Top	437277.12	6784909.29	-2493.98	3174.38
996	33/12-B-7	BCU	437991.98	6787479.12	-2414.07	2895.73
999	33/12-B-7	Cook Fm. SF Top	438034.94	6787541.18	-2507.91	3016.17
1008	33/12-B-8	BCU	438199.97	6788191.36	-2415.86	3369.04
1011	33/12-B-8	Cook Fm. SF Top	438267.78	6788294.28	-2518.26	3529.32
1036	33/9-1	BCU	437787.55	6791573.71	-2438.97	2464.10
1039	33/9-1	Cook Fm. SF Top	437791.71	6791574.06	-2642.21	2667.39
1090	33/9-4	BCU	441761.93	6800176.19	-2530.25	2555.46
1093	33/9-4	Cook Fm. SF Top	441761.93	6800179.29	-2707.66	2732.91
1124	33/9-9	BCU	441398.60	6795290.18	-2386.85	2413.00
1126	33/9-9	Cook Fm. SF Top	441393.16	6795284.94	-2475.60	2502.07
1844	33/9-A-41	BCU	437801.34	6789586.61	-2396.40	3705.86
1846	33/9-A-41	Cook Fm. SF Top	437771.38	6789411.71	-2558.19	3946.26

Figure 5.2. Preview of the selected wells and well tops

5.2 Acoustic Impendence

The next step of the process is to compute for Acoustic Impedance (AI) log. This process is carried out in Petrel. Acoustic impedance Eq. (25) is calculated from the product of compressional velocity derived from the sonic log (DT) and density derived from the density log (RHOB) and is a fundamental physical property of rocks:

$$Z = \rho v \tag{25}$$

Where;

Z is Acoustic impedance,

ρ is density,

v is compressional velocity.

5.3 Facies classification

In this process for the wells, a GR cut-off was determined that most effectively separated sand from shale interbeds. The 'sand' and 'shale' lithologies were distinguished and classed based on the Gamma Ray (GR) level. In this study, the sand was defined by a lower GR level (GR<70), while shale was defined by a higher GR level (GR>= 70). Figure 5.3 shows the process applied in Python. This step was executed using the Pandas library in Python.

Facies Classification

```
df [ 'Facies' ] = np . zeros ( len ( df ))
df [ 'Facies' ] =  np . where ( df [ 'GR (gAPI)' ] > = 70 , 1 , 0 )
combined_df [ 'Facies' ] = np . zeros ( len ( combined_df ))
combined_df [ 'Facies' ] =  np . where ( combined_df [ 'GR (gAPI)' ] > = 70 , 1 , 0 )
test [ 'Facies' ] = np . zeros ( len ( test ))
test [ 'Facies' ] =  np . where ( test [ 'GR (gAPI)' ] > = 70 , 1 , 0 )
```

```
combined_df.iloc[14:25]
```

		X (m)	Y (m)	Z (m)	MD (m)	GR (gAPI)	DT (s / ft)	RHOB (g / cm3)	Porosity (m3 / m3)	SW	PERM (mD)	LogPERM (mD)	AI	Facies
33_12_2	14	438628.870869	6.788584e + 06	-2473.0	2498	52.60	107.30	2.2665	0.2249	0.25900	14.4668	1.1604	6438.4380	0
	15	438628.870869	6.788584e + 06	-2474.0	2499	51.91	107.46	2.2843	0.2146	0.26553	12.7332	1.1049	6479.1558	0
	16	438628.870869	6.788584e + 06	-2475.0	2500	52.64	107.26	2.3316	0.1870	0.40019	5.2410	0.7194	6625.8008	0
	17	438628.870869	6.788584e + 06	-2476.0	2501	69.76	108.37	2.3965	0.1594	0.77210	0.7498	-0.1250	6740.7026	0
	18	438628.870869	6.788584e + 06	-2477.0	2502	66.64	97.43	2.3979	0.1586	0.68820	1.1394	0.0567	7509.0933	0
	19	438628.870869	6.788584e + 06	-2478.0	2503	70.69	105.43	2.3868	0.1649	0.68660	1.2407	0.0937	6900.5059	1
	20	438628.870869	6.788584e + 06	-2479.0	2504	73.38	106.51	2.4119	0.1507	0.69318	1.0052	0.0022	6902.8999	1
	21	438628.870869	6.788584e + 06	-2480.0	2505	72.40	117.00	2.3931	0.1613	0.66050	1.3377	0.1264	6234.5972	1
	22	438628.870869	6.788584e + 06	-2481.0	2506	71.97	116.18	2.3748	0.1716	0.70358	1.2409	0.0937	6231.5186	1
	23	438628.870869	6.788584e + 06	-2482.0	2507	77.56	113.61	2.3725	0.1729	0.72985	1.1069	0.0441	6365.3374	1
	24	438628.870869	6.788584e + 06	-2483.0	2508	76.74	118.83	2.3568	0.1818	0.76528	1.0138	0.0059	6045.7139	1

Figure 5.3. Dataframe after the facies classification process

5.4 Exploratory Data Analysis

Exploratory data analysis (EDA) describes the key process of analyzing and investigating data sets to summarize their main statistical features, often utilizing data visualization techniques. EDA determines the best way to manipulate the data sources to get needed answers, thereby making it easier to discover patterns, spot anomalies, test hypotheses, or check assumptions.

The statistical index of the data points fed to the ML models is shown in Table 2, and a glimpse of the data distribution of all the related datasets used in this study is displayed. It was needed to identify apparent errors, better understand patterns within the data, detect outliers or anomalous events, and find exciting relationships among the variables. Additionally, the ultimate goal of this process is to provide data insights that would inspire subsequent feature engineering and the model-building

process. The dataset shows a total count of 2298 total data points from all the total seventeen wells with a count of 2121 data points for each of the possible input featutres from the sixteen wells and 177 datapoints (porosity values) to be predictied for the seventeenth well (blind well).

Table 2. Statistics summary of the dataset

	Count	Mean	Std	Min	25%	50%	75%	Max
X(m)	2298.0	4.378611e+05	1008.117613	4.365942e+05	4.372722e+05	4.376660e+05	4.380313e+05	4.413985e+05
Y (m)	2298.0	6.790351e+06	6749.581311	6.784272e+06	6.786963e+06	6.788268e+06	6.789538e+06	6.810335e+06
Z(m)	2298.0	-2.534129e+03	172.149696	-3.140022e+03	-2.549638e+03	-2.484579e+03	-2.438230e+03	-2.365252e+03
MD(m)	2298.0	3.154446e+03	497.598006	2.409824e+03	2.594252e+03	3.128345e+03	3.532996e+03	3.976121e+03
CALI (in)	2298.0	1.043908e+01	1.851653	8.300000e+00	8.630000e+00	9.450000e+00	1.217000e+01	1.533000e+01
GR (gAPI)	2298.0	5.793180e+01	25.853489	1.720000e+01	4.094250e+01	5.275000e+01	7.036750e+01	3.974400e+02
DT(us/ft)	2298.0	1.038538e+02	8.799034	5.203000e+01	9.924250e+01	1.045750e+02	1.095600e+02	1.484100e+02
RHOB (g/cm3)	2298.0	2.248350e+00	0.157137	1.491200e+00	2.124925e+00	2.228700e+00	2.394875e+00	2.809500e+00
RT(ohm.m)	2298.0	1.952279e+02	481.651321	3.370000e-02	2.866700e+00	1.133460e+01	1.076579e+02	2.089300e+03
Por (m3/m3)	2298.0	2.328809e-01	0.080203	0.000000e+00	1.556750e-01	2.470000e-01	2.984750e-01	6.443000e-01
AI (kPa.s/m)	2298.0	6.677658e+03	1003.673426	3.168583e+03	5.950112e+03	6.529038e+03	7.293086e+03	1.627491e+04
Facies	2298.0	2.558747e-01	0.436447	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00	1.000000e+00

Figure 5.4 shows the multivariate relationships of the dataset. Before building a predictive model, it is good to understand the multivariate relationships between the variables. The multivariate analysis shown in Figure 5.4 examines the sand and shale facies and how they relate to several variables to see if one or more variables are predictive of a specific outcome. The predictive variables are independent, and the outcome is the dependent variable. Figure 5.4 was executed using the Seaborn pair plot in Python; by calculating matrix scatter plots to understand how the data parameters correlate.

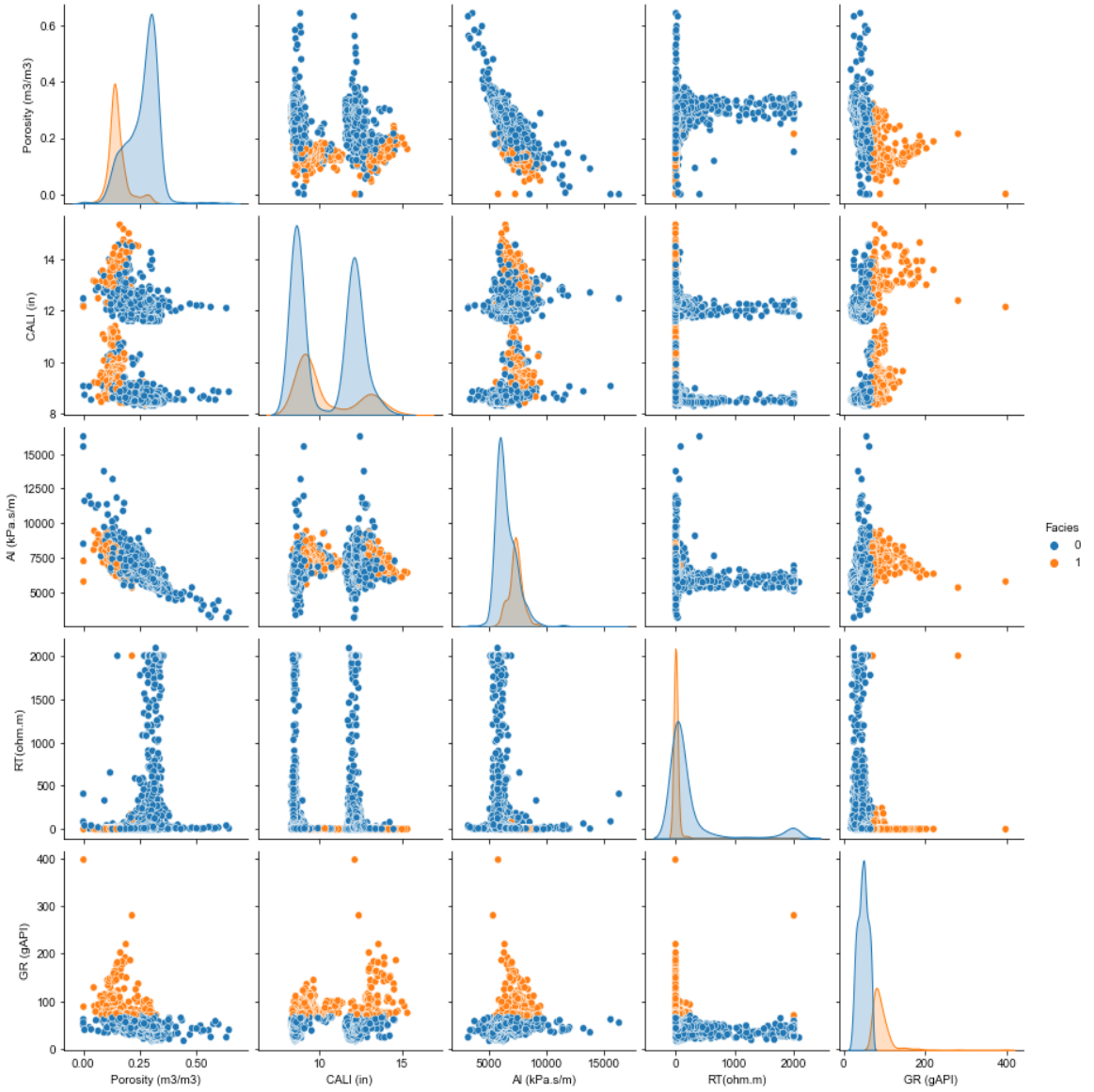


Figure 5.4. Multivariate relationships between the variables

5.5 **Label Selection**

The label selection process describes selecting the variable that one is trying to predict or forecast. In this study, porosity was selected as the label.

The porosity of a rock can be defined as the pore volume divided by the bulk volume of a rock. The porosity parameter measures how much fluid the rock can handle to hold in between the matrix grains. Porosity is dimensionless and therefore represented as a fraction between zero and one or in percentage. (Byberg, 2016)

Porosity is calculated using the relationship:

$$\phi = \frac{V_p}{V_b} = \frac{V_b - V_{matrix}}{V_b} = \frac{V_b - ({}^{W_{dry}}/_{\rho_{matrix}})}{V_b} \quad (26)$$

Where;

ϕ = porosity,

V_p = pore volume,

V_{bulk} = bulk rock volume,

V_{matrix} = volume of solid particles composing the rock matrix,

W_{dry} = total dry weight of the rock,

ρ_{matrix} = mean density of the matrix minerals.

Glover (2016) infers that porosity does not give any information concerning pore sizes, distribution, and degree of connectivity. Thus, rocks of the same porosity can have widely different physical properties. In this study, the defined porosity type used is the total porosity. The total porosity (PHIT) describes the total void space, including isolated pores and the space occupied by clay-bound water. As in Eq. (27), the bulk density and the density fluid of rock are saturated to correspond to the porosity

defined as the total porosity. Byberg (2016) describes the theoretical values for bulk density and fluid density for a sedimentary rock to range from 2.65 g/cc to 2.96 g/cc and from 1.00 g/cc to 1.4 g/cc, respectively. PHIT is calculated:

$$\phi = \frac{\rho_b - \rho_{ma}}{\rho_f - \rho_{ma}} \tag{27}$$

Where;

ϕ = Porosity,

ρ_b = mean density of the matrix minerals,

ρ_{matrix} = mean density of the matrix minerals,

ρ_f = mean density of the matrix minerals.

5.6 **Features Extraction**

From the several predictor features available, it was essential to be selective with the input variables to have a good ML models. In general, for the best prediction model, careful selection of the fewest features that provide the most amount of information is the best practice. Feature selection describes a primary process in machine learning: selecting input features for the machine learning model based on the relevance between features and model output. A good feature selection can increase the model performance with a lower error rate, and it can also enhance the model generalization and avoid overfitting problems simultaneously. The various well log features are; location (x, y, z), MD, CALI, RT, facies, GR, DT, RHOB and AI. The features are correlated with the porosity to observe its relationships and compare against each other.

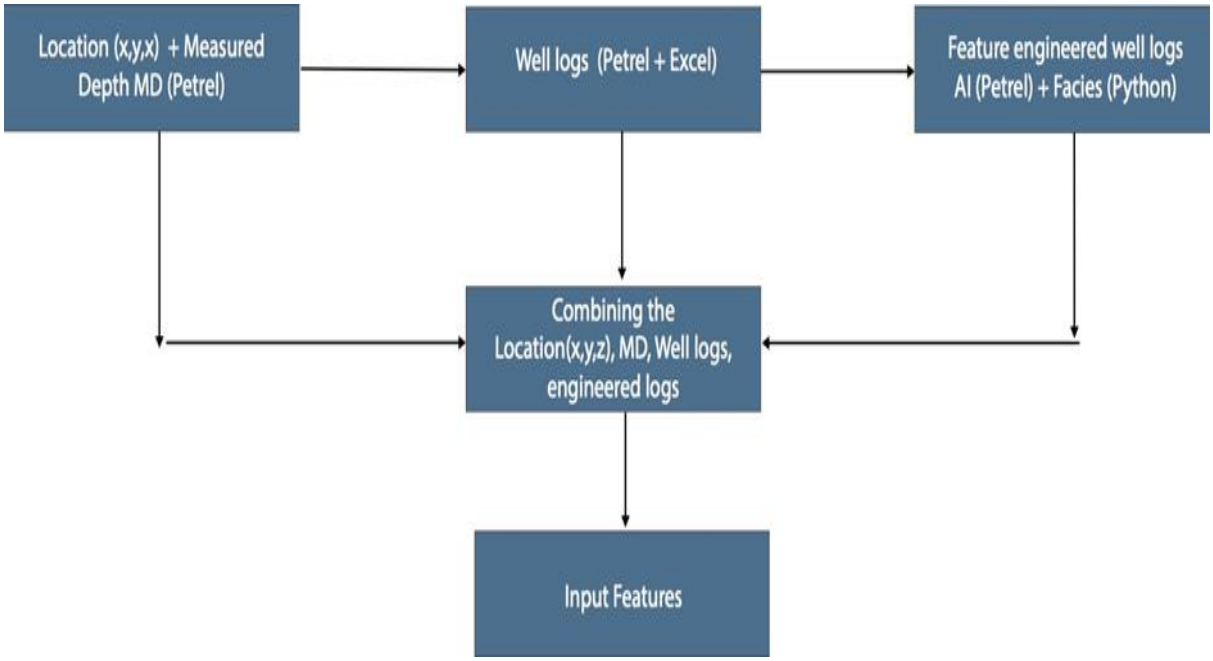


Figure 5.5. Features extraction workflow

There are two frequently used methods in feature selection; Pearson correlation and Distance correlation. The Pearson correlation was used for this study, and its concepts are discussed.

In terms of Pearson correlation, p_j represents a value in the range of +1 and -1 considering with the given dataset $\{(x_1, y_1), \ldots, (x_n x_n)\}$ by using Eq. (27). The equation shows the correlations between x and y, where +1 refers to total positive correlation and -1 refers to total negative correlation. Therefore, when the absolute value of the correlation coefficient is closer to 1, it indicates a higher correlation relationship between variables. In Pearson correlation, the relationship is measured by the absolute values of p_j. This means a higher absolute value suggests a higher correlation between the dependent variable x and y. The different sign of p_j shows whether the dependent variable y would follow the changes of the increase or decrease of x. This correlation coefficient was calculated and plotted using the seaborn library in Python. Table 8 highlights the interpretation of the Pearson correlation coefficient value. The Pearson correlation coefficient is given by:

$$p_j = \frac{\sum_{i=1}^{n}(x_{j,i} - \bar{x}_j)(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_{j,i} - \bar{x}_j)^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}, j = 1, \ldots n \quad (28)$$

Where;

p_j = correlation coefficient,

$x_{j,i}$ = values of the x-variable in a sample,

$\bar{x}_j$ = mean of the values of the x-variable,

y_i = values of the y-variable in a sample,

$\bar{y}$ = mean of the values of the y-variable.

Table 3. Interpretation of Correlation coefficient values

Correlation coefficient value	Interpretation
±1	Perfect positive/negative relationship
±0.8	Reasonably strong positive/negative relationship
±0.6	Moderate, strong positive/negative relationship
0	No relationship

The various features are loaded into the seven ML models in the following stages below;

1. Location, MD, RHOB, CALI, RT, DT, Facies, GR, AI
2. Location, MD, RHOB, RT, DT, Facies, GR, AI
3. Location, MD, RHOB, DT, Facies, GR, AI
4. Location, MD, RHOB, Facies, GR, AI

5. Location, MD, RHOB, GR, AI
6. Location, MD, RHOB
7. Location, MD, CALI, RT, DT, Facies, GR, AI

These input features are compared to see how they either enhanced or reduced the prediction accuracy for porosity. The computed porosity predictions and performance evaluation of the ML models are discussed in the next chapter.

5.7 Feature transformation

Features transformation is described as the process of modifying the dataset but keeping the information. Generally, due to the varying datasets used in this study, these modifications will make the various machine learning algorithms understanding easier, which will deliver a better result for this study.

There are many reasons to perform features transformation for the dataset, and a few of them are:

- They make the features consistent for visualization and comparison (Comparison between AI and porosity)
- To avoid bias or impose feature weighting for methods (e.g., k nearest neighbour regression) that rely on distances calculated in predictor feature space.
- The method requires the variables to have a specific range or distribution, such as
    1. Artificial neural networks may require all features to range from [-1,1]
    2. Partial correlation coefficients require a Gaussian distribution.
    3. Statistical tests require a specific distribution.
    4. A sequential geostatistical simulation to be performed requires an indicator or gaussian transform.

5.7.1 Normalization

Normalization is a data preparation technique applied for machine learning and is the feature transformation carried out in this study. It changes the values of the dataset's numeric columns to a standard scale without distorting differences in the ranges of values. Normalization transforms the feature distribution to a min of 0 and a max of 1 (-1 to +1). This process is typically a shift, and stretch/squeeze of the original property distribution assumes no shape change.

Motivations for normalization include:

- Remove the effect of the scale of different type of data (i.e., the acoustic impedance varies between 3000~12500, but porosity only varies between 0 ~0.6)
- Activation functions in neural networks have greater sensitivity when the value of nodes are closer to 0.0 (i.e., results in higher gradient and improves backpropagation in training)

Normalization is done in Python through back transformation, keeping the min and max values.

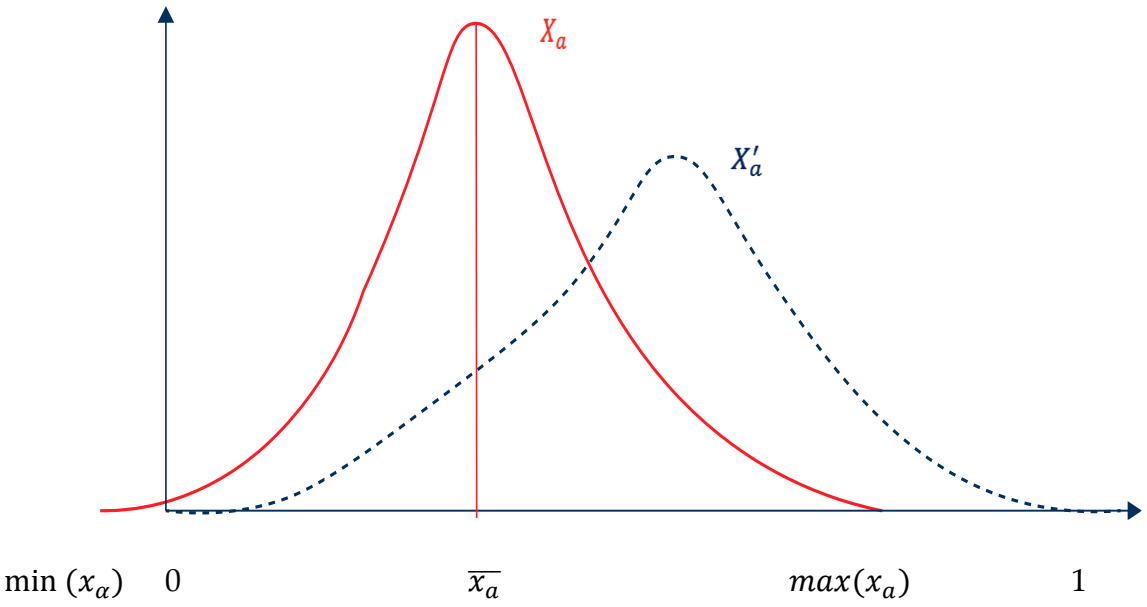


Figure 5.6. Example of normalization of a feature distribution

$$x'_a = \frac{x_a - \min(x_a)}{\max(x_a) - \min(x_a)} \tag{29}$$

5.8 Machine Learning Models Generation

The training and testing process is a fundamental process that affects an ML model's success. An effective training process significantly improves the quality of the developed system (Figure 5.7). In this study, the selected input features from sixteen wells penetrating the target reservoir interval were used to predict for the porosity in well 33/9-4 (blind well) in the field.

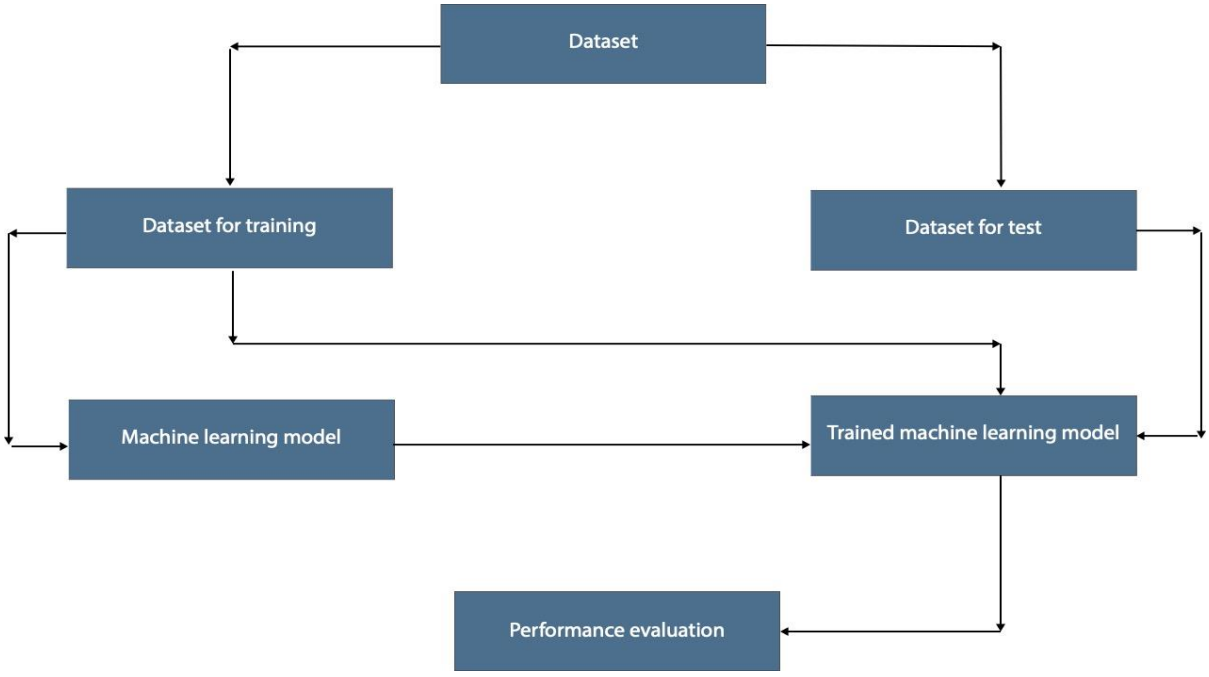


Figure 5.7. Training and testing process workflow

The set of data that enables the training is called the "training set" and in this case, the input features from the sixteen wells in the field. The same data set is processed in the training process many times as the connection weights are refined and then used to predict the "test set" which is the porosity of the blind well.

The training and test process is as follows:

- Load the normalized input feature dataset from the wells.
- Train the models with these input features.
- With the established models, run the test dataset of the blind well to evaluate the model performance.

5.9 **Performance Evaluation**

After the train and test of the various ML models, it was meaningful to evaluate several statistical measures of prediction accuracy. A few statistical prediction accuracy metrics are computed to observe the results of the porosity predictions from the compiled input features. The purpose of this process is to determine how much data is assimilated.

Performance evaluation procedures are handled according to some specific criteria which vary according to the structure of the data. Equations 29 - 31 break down these well-established and widely used statistical measures of prediction accuracy (Hyndman and Koehler, 2006).

1. R^2 computes how much better the regression line fits the data than the mean line. It defines the percentage of the change in the total change that the regression model can explain Eq. (29). Conclusively, it indicates that the independent variable can explain the percentage of total changes in the dependent variable and is given as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \quad (30)$$

Where;

y_i is the true value,

$\hat{y}_i$ is the predicted value of the i-th sample,

n is the total number of samples.

2. MAE is calculated by the sum of the square of prediction error, which is real output minus predicted output, and then divide by the number of data points. It gives an absolute number on how much the predicted results deviate from the actual number.

The best value of MAE is 0, which indicates a perfect prediction result. MAE is expressed by;

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (31)$$

3. RMSE is the square root of MSE. As MSE and RMSE approach zero, the error rate decreases Eq. (31). This is calculated as the standard deviation of the residuals (prediction errors). Residuals measure the distance of the regression line to the data points; RMSE measures how to spread out these residuals. It shows how concentrated the data is around the line of best fit. Often, the RMSE is preferred to the MSE as it is on the same scale as the data. Historically, the RMSE and MSE have been popularly used, mainly because of their theoretical relevance in statistical modeling (Hyndmana, 2006). RMSE is expressed by:

$$RMSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (32)$$

6 Results

6.1 Features Evaluation

Generally, model performance is highly influenced by the input features used for training and validation, and it is essential to find out which features are the most relevant to predict porosity. Using the right features is essential as poor input features lead to redundancy and reduced model interpretability. In this study, the correlation between various features and porosity was measured by Pearson correlation coefficient (p), with higher p values ~ $\geq \pm 0.6$ representing the features showing a moderately strong relationship to porosity. The features fed in stages into the ML models (as in chapter 5) are compared as they are gradually reduced one by one based on the value of their Pearson correlation. Therefore, the features with low correlation coefficient values are removed first. From the Pearson correlation in Eq. (28), four moderate to strong features are identified: RHOB, AI, facies, and GR logs; while those with less correlation with porosity are; CALI, RT and DT, as Figure 6.1 shows.

RHOB and AI show the strongest correlation p ~ -0.98 and p ~ -0.8, respectively. Both the GR and facies correlation value shows a moderately strong correlation of p ~ -0.6. Although the facies is generated from classifying the GR log values, it is observed that the facies showed a positive correlation p ~ 0.8 with the GR. The negative correlation between facies and GR to porosity indicates that high GR values indicate less porous rock space for shale, where the porosity is remarkably low.

The CALI and RT, although significant when identifying hydrocarbon pay zones, shows to have a weak correlation with porosity, p ~ -0.1 and p ~ 0.4, respectively, as shown in Figure 6.1. These two features show the weakest correlation as they are below $p \geq (\pm 0.6)$.

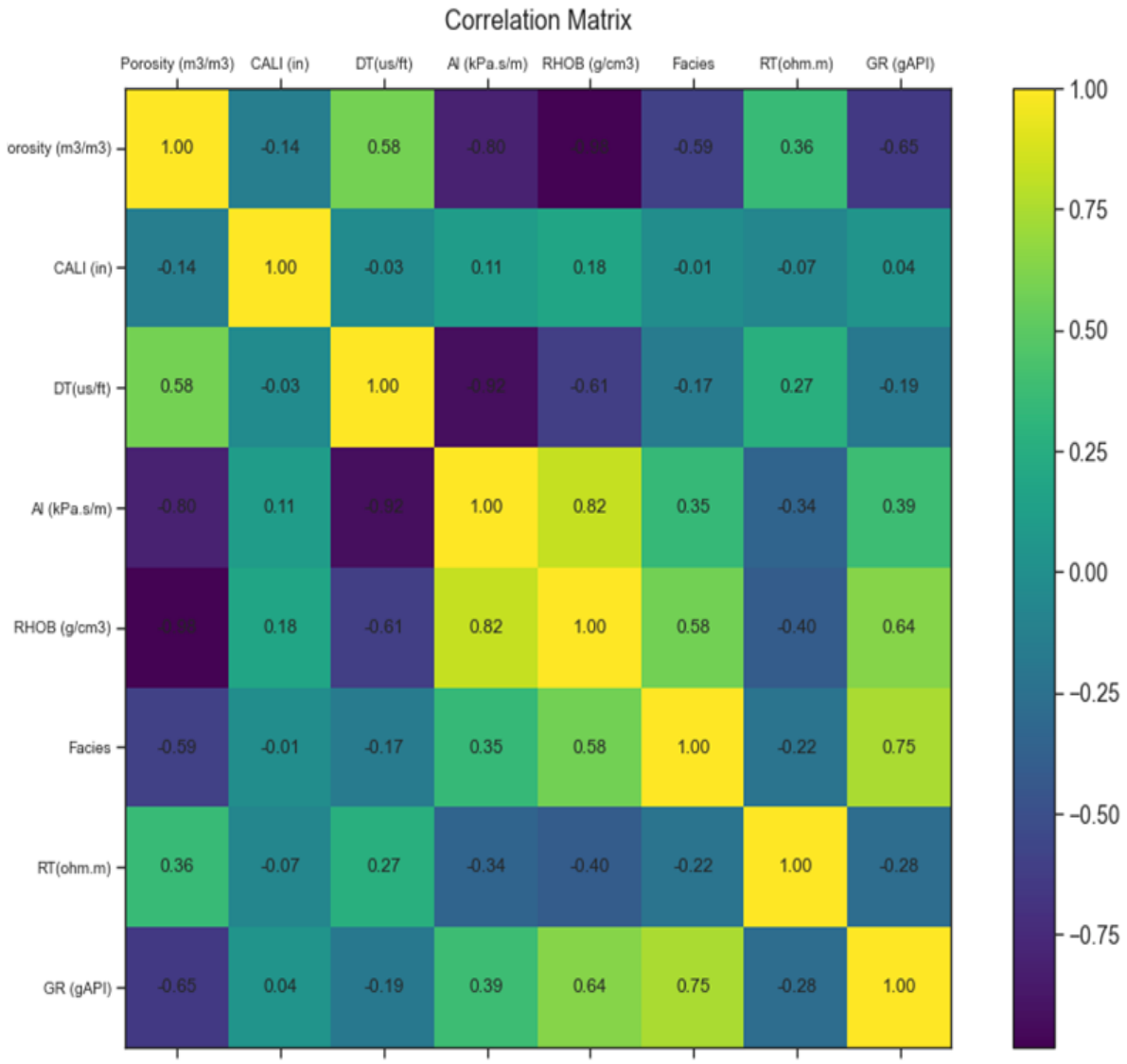


Figure 6.1. Pearson correlation matrix

6.1.1 Prediction results

This section visually shows the results of the blind well (33/9-4) predictions to compare and evaluate the prediction performance of the models using the various features. Figure 6.2 and Figure 6.3 are introduced to compare the results of the ML models visually. Similarly, Figure 6.4 and Figure 6.5 show the predicted porosity of the blind well in MD. Visually the BRR, LR and RF show the closest match between the predicted and actual porosity log and are identified as the best performing ML

models. The moderately performing ML models are the NN and DT, while the SVM and KNN ML models show the weakest predictions, with the predicted porosity values lagging behind the actual values in the reservoir interval of the well.

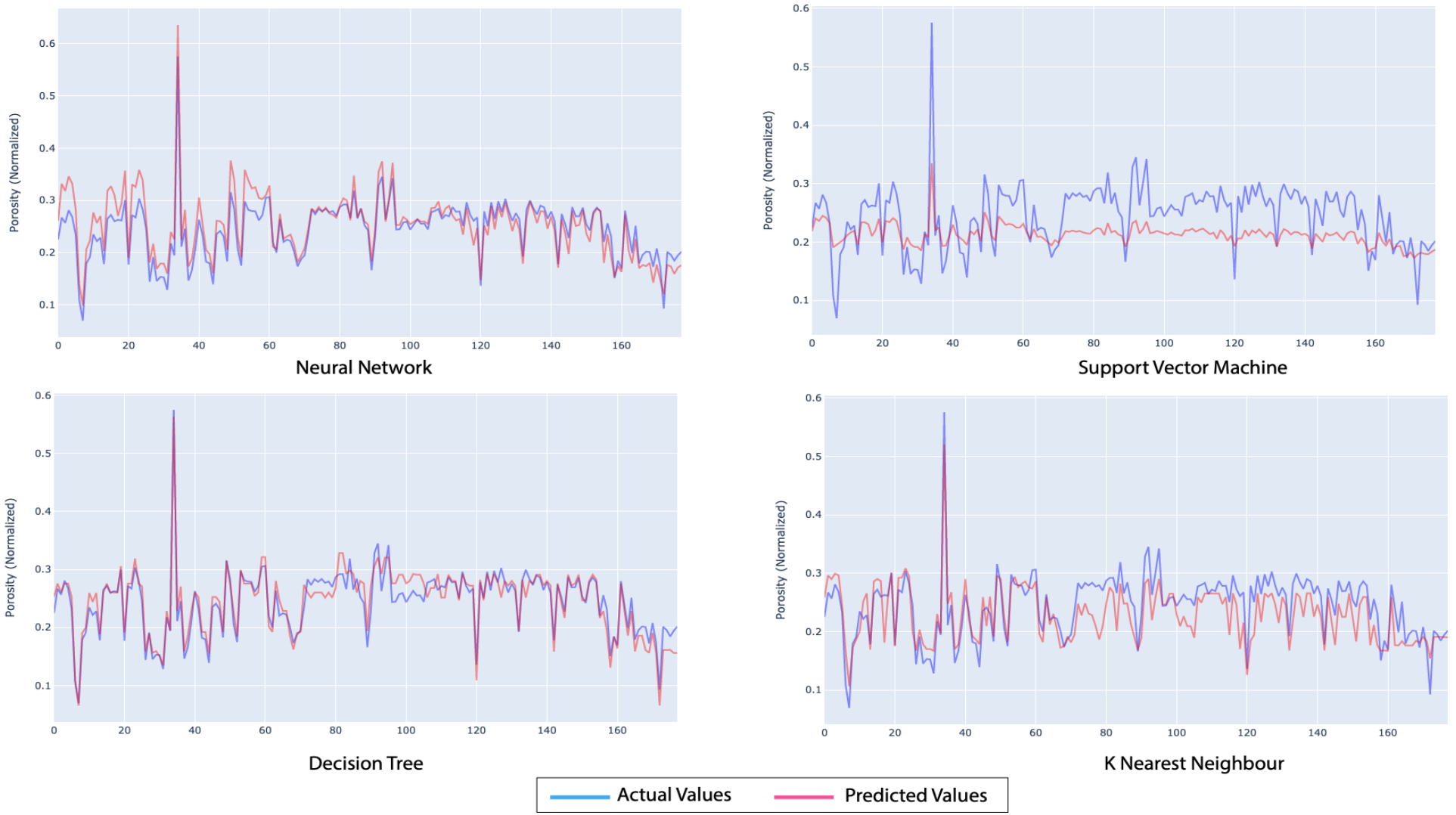


Figure 6.2. Predicted vs. actual porosity values using all input features for the NN, SVM, DT and KNN machine learning models

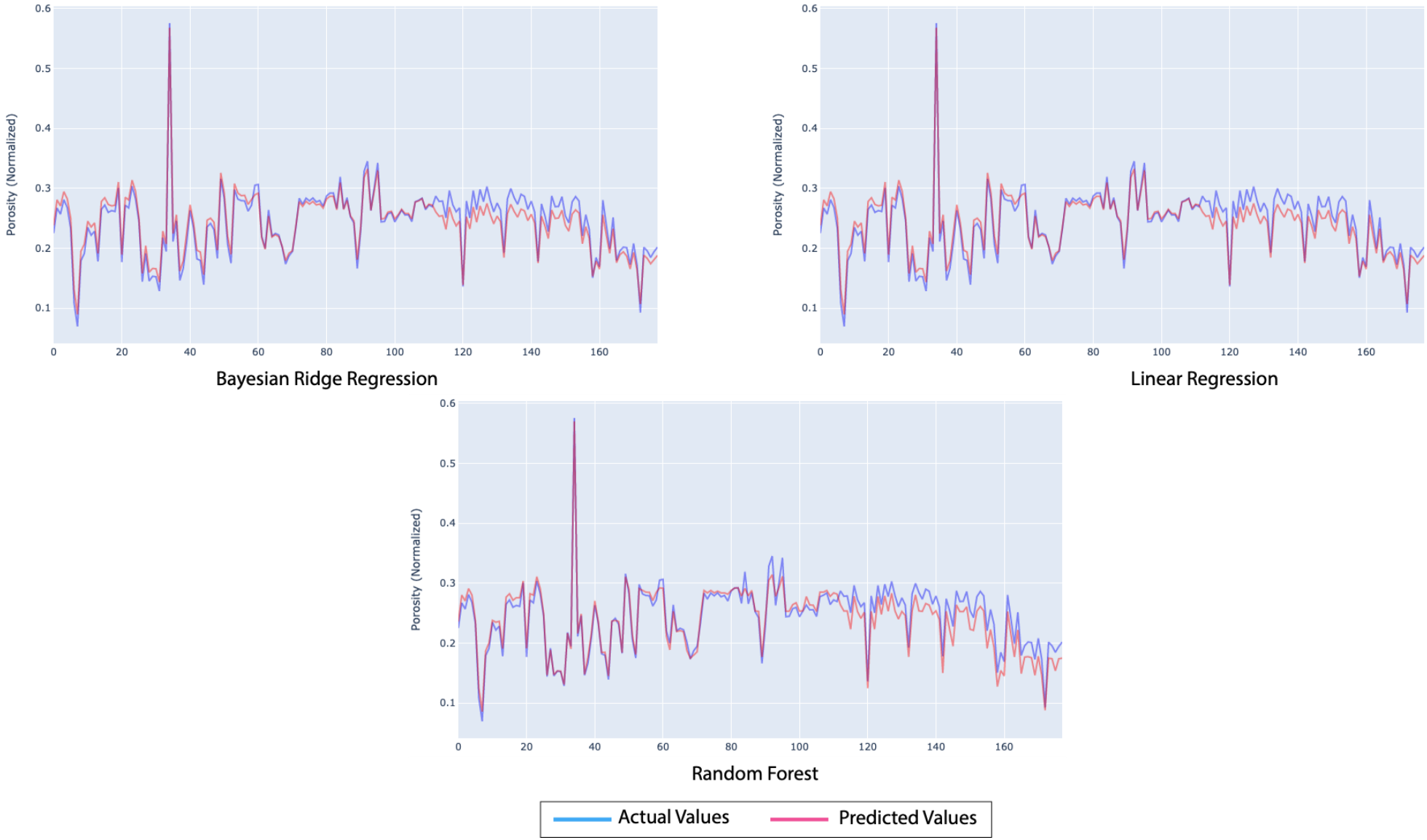


Figure 6.3. Predicted vs. actual porosity values using all input features for the BRR, LR and RF machine learning models

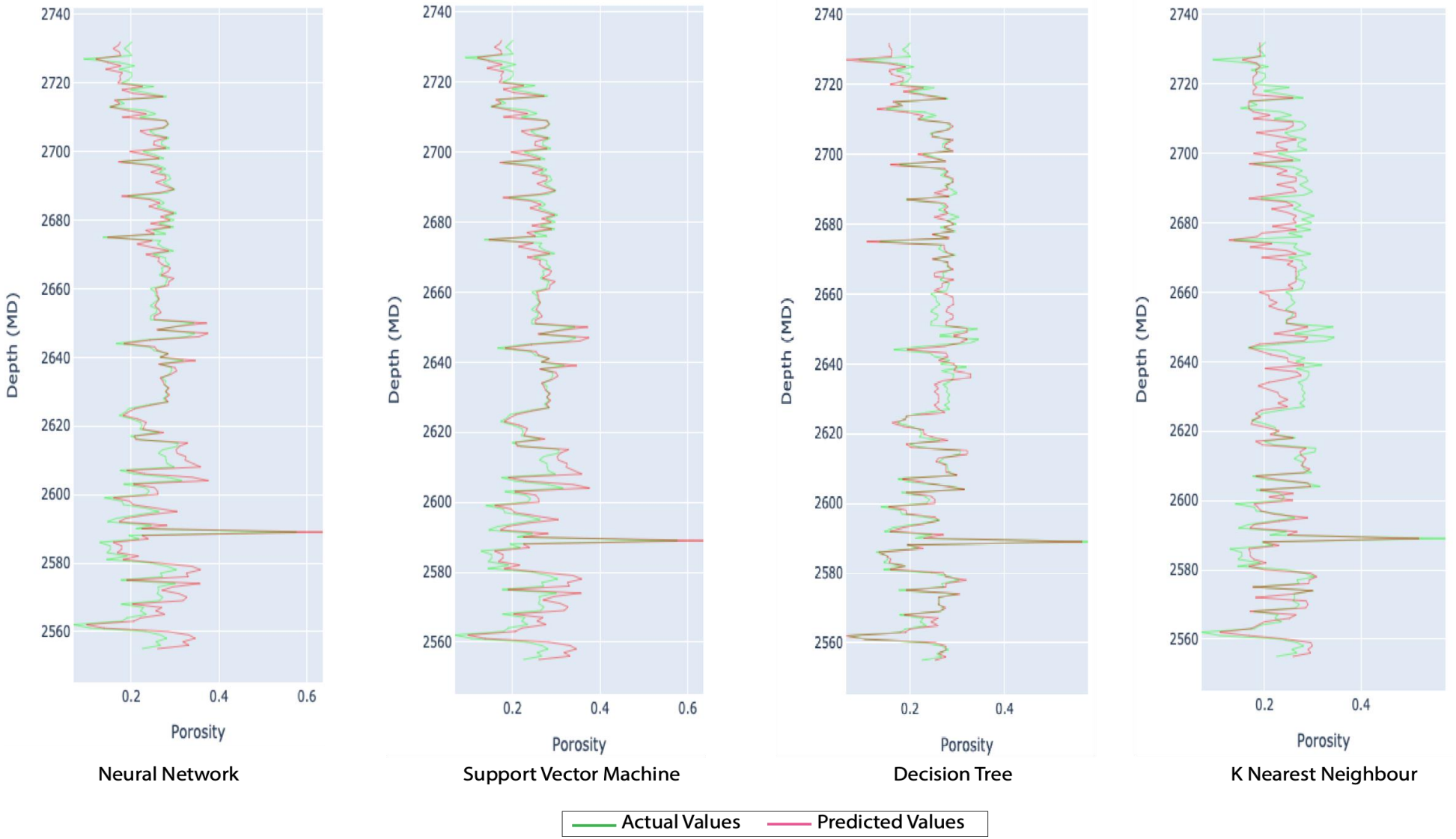


Figure 6.4. Predicted and actual porosity log in the blind well using all input features for NN, SVM, DT and KNN machine learning models

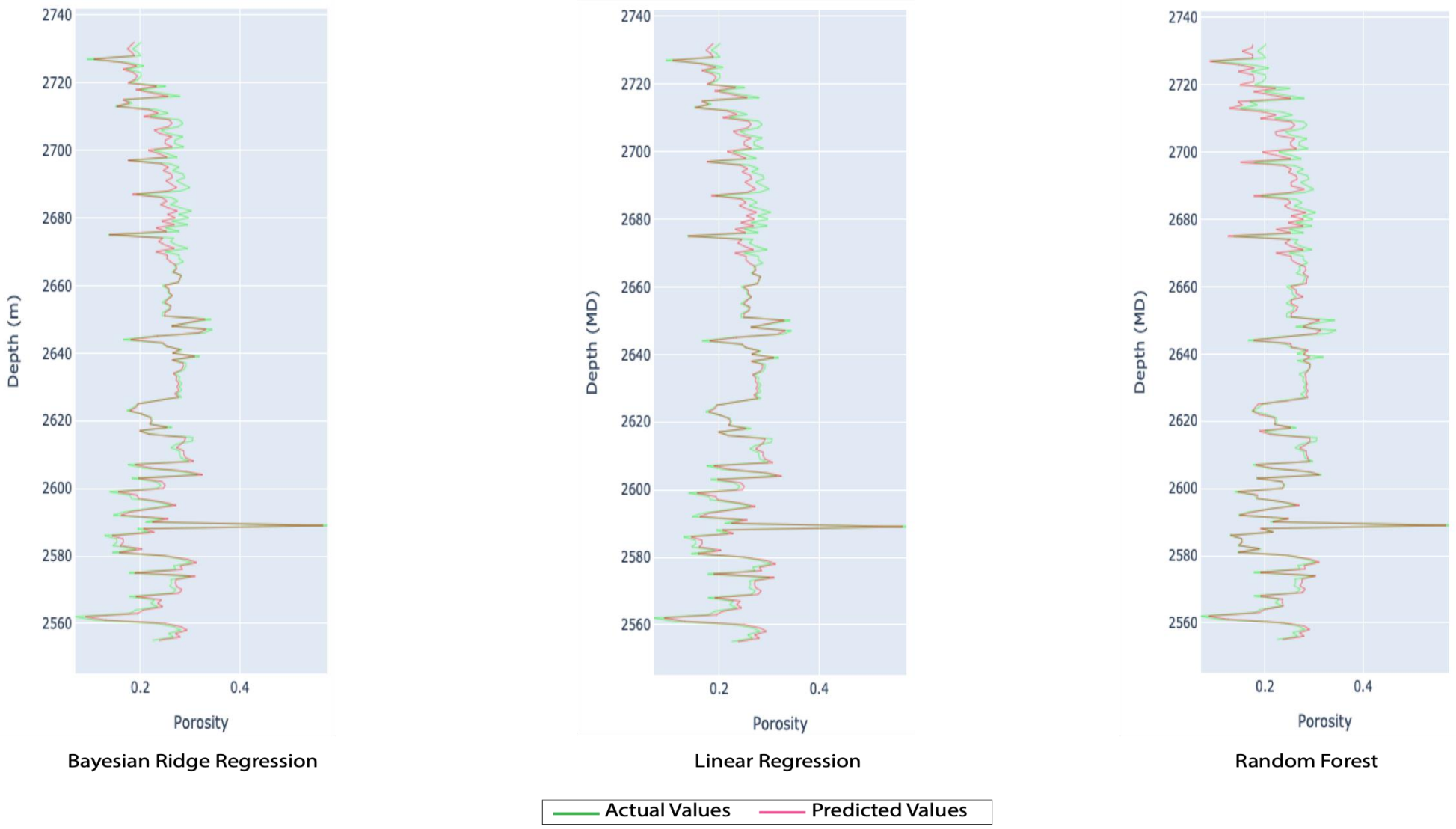


Figure 6.5. Predicted and actual porosity log in the blind well using all input features for LR, BRR and RF machine learning models

Table 4. R^2 performance evaluation summary using different well log inputs

ML Models	RHOB, CALI, RT, DT, Facies, GR, AI	RHOB, RT, DT, Facies, GR, AI	RHOB, DT, Facies, GR, AI	RHOB, Facies, GR, AI	RHOB, GR, AI	RHOB, AI	RHOB	CALI, RT, DT, Facies, GR, AI
NN	0.765	0.817	0.791	0.625	0.823	0.761	-13.770	0.725
SVM	0.114	0.089	0.125	0.115	0.124	0.146	0.217	0.052
DT	0.874	0.860	0.677	0.670	0.663	0.691	0.721	0.597
KNN	0.623	0.584	0.595	0.776	0.774	0.739	0.827	-0.306
BRR	0.935	0.931	0.931	0.933	0.933	0.933	0.934	0.718
LR	0.935	0.931	0.931	0.933	0.933	0.933	0.934	0.719
RF	0.907	0.910	0.872	0.873	0.873	0.872	0.875	0.751

Table 5. RMSE performance evaluation summary using different well log inputs

ML Models	RHOB, CALI, RT, DT, Facies, GR, AI	RHOB, RT, DT, Facies, GR, AI	RHOB, DT, Facies, GR, AI	RHOB, Facies, GR, AI	RHOB, GR, AI	RHOB, AI	RHOB	CALI, RT, DT, Facies, GR, AI
NN	0.084	0.077	0.078	0.112	0.082	0.076	0.218	0.080
SVM	0.052	0.053	0.052	0.052	0.052	0.051	0.049	0.054
DT	0.021	0.020	0.033	0.033	0.032	0.032	0.031	0.033
KNN	0.034	0.036	0.035	0.026	0.026	0.028	0.023	0.063
BRR	0.014	0.015	0.014	0.014	0.014	0.014	0.014	0.029
LR	0.014	0.015	0.014	0.014	0.014	0.014	0.014	0.029
RF	0.017	0.017	0.020	0.020	0.020	0.020	0.020	0.028

Table 6. MAE performance evaluation summary using different well log inputs

ML Models	RHOB, CALI, RT, DT, Facies, GR, AI	RHOB, RT, DT, Facies, GR, AI	RHOB, DT, Facies, GR, AI	RHOB, Facies, GR, AI	RHOB, GR, AI	RHOB, AI	RHOB	CALI, RT, DT, Facies, GR, AI
NN	0.063	0.056	0.060	0.095	0.061	0.057	0.210	0.059
SVM	0.043	0.044	0.043	0.043	0.042	0.041	0.036	0.045
DT	0.016	0.016	0.026	0.026	0.026	0.025	0.024	0.025
KNN	0.028	0.029	0.029	0.021	0.022	0.024	0.018	0.050
BRR	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.019
LR	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.019
RF	0.014	0.014	0.016	0.016	0.016	0.016	0.016	0.021

6.1.2 Performance evaluation

Table 4, 5, and 6 showing the performance evaluation highlight the ML model prediction performance summary using a varying number of input features. With increasing additional features into the ML models, a few models showed improved performance metrics, and for some of the others, there was a considerable drop in their performance.

The tables show that the best performing ML models with changing input features remain the BRR, LR and RF. Both BRR and LR have the highest R^2 ~ 0.933, lowest RMSE ~ 0.014, and MAE ~ 0.012 when the RHOB is an input feature in the model. However, when RHOB is not included as an input feature, the two ML model's performances range the same with R^2 ~ 0.718 for BRR and R^2 ~ 0.719 for LR with common values of RMSE and MAE to be ~ 0.029 and ~ 0.019 respectively.

The RF also performed highly with R^2 ~ 0. 907, RMSE ~ 0.017, and MAE ~ 0.014 when all the input features are used and also has the highest R^2 ~ 0.751, lowest RMSE ~ 0.028, and MAE ~ 0.021 without the feature RHOB. The average performing models are the DT and the KNN with their R^2, RMSE and MAE varying as the input features are reduced and perform poorly without the RHOB feature.

Ostensibly, the values of the NN show a relationship not matching with other values, with varying inputs the model R^2 value changes were performing inconsistently as to the other ML models. The NN also consistently showed the highest RMSE and MAE with the changing inputs, highlighting its somewhat poor performance in this study.

At the very least in the performance metrics is the SVM, as seen in Table 4, Table 5, Table 6, having a high variance between the predicted and the actual values. When compared to other models, SVM shows poor performance; having the lowest R^2 ~ 0. 052, highest RMSE ~ 0. 052 and MAE ~ 0.043 when all features are inputted into the models.

7 Discussions

In order to interpret the various ML model results, the porosity estimation performances need to be analyzed. In this chapter, the results are discussed and interpreted based on the performance evaluation of the various ML models used in this study.

The BRR and LR were the best-performing models in this study. These algorithms had a range of $R^2 \sim 0.93$, RMSE ~ 0.014 and MAE ~ 0.012 respectively. These ML models showed an improvement in the performance of the models when new features are involved, as the improved data range also impacted their performance, which led to its improved accuracy. The LR, although susceptible to overfitting, showed higher accuracy than other models as the algorithm learns to train from the understanding of the various linear relationships of the input features and hence, its strong prediction accuracy. The excellent BRR model performance can be attributed to the model viewing the porosity estimation as a problem of integrating prior information with information gained from the input features, formalized using probability distributions as it does not treat regression as an optimization problem.

The study showed the RF also having good estimation results as it has the highest R^2 ~ 0. 751, the lowest MSE ~ 0.028 and MAE ~ 0.021 compared to the other ML models without the RHOB feature. The RF achieved this performance due to its ensemble tree bagging of the multiple data inputs. The ensemble reduced the model variance as expected, and due to a large number of data points, its ensemble algorithm approach reduced the risk of overfitting. Also observed is that the RF could utilize the new information from the added features to improve the overall result of its prediction, unlike the SVM and KNN.

However, The NN achieved a considerable moderate to high R^2 with strongly correlated values, as shown in Table 4 with the NN model having a high R^2 and a

corresponding higher RMSE. However, a drawback of R^2 is it can only increase as more predictors are added to the regression model. This increase is artificial when predictors are not improving the model's fit (Martin, 2020). Since this is a regression problem the R^2 is what is given priority over the RSME when accessing the model performance. As the R^2 is a relative measure of fit and the RMSE an absolute measure of fit.

In this study, the DT results show a poor to average performance with the varying input features. In most studies, the DT shows to be less effective in predicting the outcome of the continuous variables, as in this case, porosity. Unlike the RF, which has the power to handle large data sets with higher dimensionality. The weak performance indicates that the DT tends to lose information and is entirely inadequate when applied to regression. However, another reason for its subpar performances may be attributed to the small continuous change in the data, which causes a significant shift in the DT structure, thereby causing instability. The performance metrics of the DT were significantly affected by the new feature and this can be attributed to overfitting as it has learned a meaningless pattern out of random noise created by the additional feature and hence, low information. Reducing the number of input features increased the prediction accuracy, as observed in Table 4. However, the model's generalizability can be limited because the model performance would only rely on the data from few petrophysical logs.

Conclusively, the BRR, LR and RF provide more accuracy and advantages over the other models in predicting porosity. This indicates that these ML models can be applied to more wells with an extensive data range but might slightly be less accurate.

The advantages are but not limited to:

- The models provide a more efficient and economical method to obtain reservoir porosity against conducting coring analysis, which is widely used to acquire

petrophysical properties from reservoir formations or intervals interest accurately.

- Reduce the cost of well-logging services because some well logs used for porosity estimation would become unneeded in the future as exploration companies aim to reduce operating costs.

As for the limitations experienced by the ML models, the generalizability of the various models are limited by the training and validation dataset. The porosity prediction accuracy of the blind well performs decently in a few and excellent in others, but it may not have the same performance for another well from fields close to each other with the same data in training and validation due to heterogeneity. The generalizability of applying the BRR, LR and RF models in other wells in the Statfjord field can be enhanced by increasing the number of wells used in training and validation. These three ML models can predict reservoir porosity in the North Sea by introducing more wells from various fields within the North Sea.

8 Conclusion

In this study, several machine learning algorithms are applied for the porosity prediction in a blind well (33/9-4) penetrating the reservoir interval of the Brent Group to Top Cook located in the Statfjord field. The purpose was to compare the various ML models and find the best models for future porosity predictions as the oil and gas industry transforms to finding data-driven solutions in carrying out its operations.

The Bayesian ridge regression (BRR), linear regression (LR) and the random forest (RF) presented the best porosity estimation result. This result could be further improved if more datasets from more wells penetrating the reservoir target in the field were inputted into the model. Overall it shows a potential to provide an innovative method for petrophysical evaluation in the oil and gas industry

The density (RHOB) ~ -0.98 and acoustic impedance (AI) ~ -0.8 are the highest correlating features to porosity, as shown in the Pearson correlation matrix. By feature engineering the GR logs, facies logs are generated by applying cut-off values of GR >= 70 as shale and GR < 70 as sand. In this study, the facies showed a moderately strong correlation (~ -0.6); with its addition as an input feature, it improved the porosity estimation of some of the ML models. As sand facies can be related to higher porosity and shale facies to lower porosity, a relationship is established, significantly impacting generating more training data for the models leading to increased prediction accuracy.

Conclusively, ML retrieves predictive patterns from data, and the quality and quantity of data are crucial for successful predictive accuracy. Predictive patterns have similar trends for different machine learning models, but they are still different in detail. This indicates that machine learning works differently and can be applied for predicting petrophysical properties in the broader North Sea. Adequate data with low noise are necessary for ML models to recall reliable predictive patterns. An adequate

dataset should be sufficiently representative as if the dataset is too small to be representative; ML models may learn only local predictive patterns. As in this study, 2298 data points, which is not an adequate dataset, but representative enough for the Statfjord field. These 2298 samples from the selected wells represent the Brent Group of the Statfjord field, while other fields can be further explored and analyzed. ML studies with a larger dataset will ensure that the oil and gas industry's routine core analysis and logging operations would be reduced and make exploration and production activities more cost-effective.

9 Future Work Recommendations

Some work recommendations to be considered in future studies include:

- Increasing the training and validation dataset by utilizing more well logs from wells penetrating potential reservoir targets of interest to build a more representative model to make porosity predictions for those targets.
- Adding more valuable information from horizons into the model so that ML models can be built using data from the combination of well logs and horizons.
- Using similar or advanced ML algorithms to predict for other important petrophysical properties and lithofacies.
- Generate a 3D acoustic impedance model from seismic reflection amplitudes to serve as an input feature.

10 References

Akinnikawe, O., Lyne, S., & Roberts, J. (2018). Synthetic Well Log Generation Using Machine Learning Techniques. Paper presented at the SPE/AAPG/SEG Unconventional Resources Technology Conference, Houston, Texas, USA.

Al Khalifah, H; Glover, P.W.J; Lorinczi, P (2020). Permeability prediction and diagenesis in tight carbonates using machine learning techniques Marine and petroleum geology, 2020-02, Vol.112, pp.2-4.

Alpaydin Ethem. Introduction to Machine Learning, 2014. Introduction to Machine Learning (3rd Edition) p.3.

Al-Anazi A. F., Gates I. D. (2015) On Support Vector Regression to Predict Poisson's Ratio and Young's Modulus of Reservoir Rock in Artificial Intelligent Approaches in Petroleum Geosciences et al., AG, 2015. 1: 167 -190.

Ashena Rahman, Thonhauser Gerhard, (2015). Application of Artificial Neural Networks in Geoscience and Petroleum Industry. pp 127-166

Atle Aadland ; Olav Dyrnes ; S.R. Olsen ; O.M. Drønen. "Statfjord Field: Field and Reservoir Management Perspectives." SPE Res Eng 9 (1994): 157–161. DOI: https://doi.org/10.2118/25027-PA.

Bhattacharya Shuvajit, and Srikanta Mishra. (2018) "Applications of machine learning for facies and fracture prediction using Bayesian Network Theory and Random Forest: Case studies from the Appalachian basin, USA." Journal of Petroleum Science and Engineering 170: 1005-1017.

Boyle Brandon H., (2011). Support Vector Machines: Data Analysis, Machine Learning and Applications, p. 2.

Bronshtein Adi, (2017). A Quick Introduction to K-Nearest Neighbours Algorithm. https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7.

Byberg Ingrid, (2016). Reservoir Characterization of the Skagerrak Formation in the Central North Sea, 1, p. 44.

Buland Arild, Seismic Analysis and Inversion, (2020). GEO 670, University of Stavanger p. 299.

Cranganu C. (2015). Artificial Intfelligent Approaches in Petroleum Geosciences, DOI 10.1007/978-3-319-16531-8_7.

Dell'Aversana, Paolo. (2019). Comparison of different machine learning algorithms for lithofacies classification from well logs. Bollettino di Geofisica Teorica ed Applicata pp 69 – 80.

Deegan C. E, and Scull B, J (1977). A proposed standard lithostratigraphic nomenclature for the Central and Northern North Sea. Report of the Institute of Geological Sciences, No. 77/25; Bulletin of the Norwegian Petroleum Directorate, No. 1.

Deng Chengxiang, Heping Pan, Ahmed Amara Konaté, Sinan Fang, (2017) Support vector machine as an alternative method for lithology classification of crystalline rocks DOI:10.1088/1742-2140/aa5b5b

Duffy Oliver B, (2015). The growth of non-collinear normal fault systems; What can we learn from 3D seismic reflection data? p. 5.

Evans, D., Graham, C., Armour A. and Bathurst, P (eds), (2003). The Millenium Atlas: Petroleum Geology of the Central and Northern North Sea: Published by The Geological Society of London. 1, pp. 290-291.

Gibbons K. A., C. A. Jourdan and J. Hesthammer (2003). The Statfjord Field, Blocks 33/9, 33/12 Norwegian sector, Blocks 211/24, 211/25 UK sector, Northern North Sea Geological Society, London, Memoirs, 20, pp. 335-353 .

Glover, Paul WJ, (2000). "Petrophysics MSc Course Notes." University of Aberdeen, UK.

Hall, B. (2016). Facies classification using machine learning. The Leading Edge, 35(10), 906-909. doi:10.1190/tle35100906.1.

Hyndman Rob J., Koehler Anne B. , (2006). Another look at measures of forecast accuracy https://doi.org/10.1016/j.ijforecast.2006.03.001.

Latimer, R. B., Davidson, R., & Van Riel, P. (2000). An interpreter's guide to understanding and working with seismic-derived acoustic impedance data. The leading edge, 19(3), pp. 242- 256.

Martin Karen Grace (2020). Assessing the Fit of Regression Models https://www.theanalysisfactor.com/assessing-the-fit-of-regression models/#:~:text=The%20RMSE%20is%20the%20square,an%20absolute%20measure%20of%20fit.&text=Lower%20values%20of%20RMSE%20indicate%20better%20fit.

Mohammadlou M.H., Mørk M.B (2012). Integrated permeability analysis in tight and brecciated carbonate reservoir SPE Reservoir Eval. Eng., 15 (6), pp. 624-635.

Mondol, N, H., (2003). Reservoir prediction through 3D seismic interpretation and post-stack seismic inversion – An integrating approach to reveal reservoir properties in Gullveig Field, northern North Sea. Master dissertation at Norwegian University of Science and Technology, pp 32-36.

Montgomery, Douglas C, Peck A Elizabeth, and Vining G Geoffrey, (2012). Introduction to Linear Regression Analysis, pp 1-5.

Murphy, Kevin P, (2012) Machine Learning: A Probabilistic Perspective. p. 1.

Nikhil Buduma (2017). Fundamentals of deep learning: designing next-generation machine intelligence algorithms p. 3.

NPD, (2021). The Norwegian North Sea: Geology of the North Sea Https://www.npd.no/en/facts/publications/co2-atlases/co2-atlas-for-the-norwegian-continental-shelf/4-the-norwegian-north-sea/4.1-geology-of-the-north-sea/.

Pasanen Leena, Holmstrom Lasse and Sillanpaa Mikko J. (2015). Supporting Information for 'Bayesian LASSO, scale space and decision making in association genetics. p.1.

Pyrcz Michael, Support Vector Machine, Petroleum Geostatitics, Teaching Notes, University of Texas at Austin (2020). https://www.youtube.com/watch?v=UpN6TLMJiGg&t=786s.

Rahman Ashena and Gerhard Thonhauser (2015). Application of Artificial Neural Networks in Geoscience and Petroleum Industry in Artificial Intelligent Approach in Petroleum Geosciences, edited by Constantin Cranganu et al., AG, 1: p. 128.

Spencer A.M, Karlsson W., (1987). Habitat of Hydrocarbons on the Norwegian continental shelf: The Snorre, Statfjord and Gullfaks oilfields and the habitat of hydrocarbons on the Tampen Spur, offshore Norway 1, pp. 181-187.

Sun Jian, Zhang Rongjun, Chen Mingqiang, Chen Bo, Wang Xiao, Li Qi & Ren Long (2021). Identification of Porosity and Permeability While Drilling Based on Machine Learning. pp. 2-7.

Vapnik, Vladimir, Steven E. Golowich, and Alex J. Smola. (1995). "Support vector method for function approximation, regression estimation and signal processing." Advances in neural information processing systems. pp. 281-287.

Yasin Qamar, Sohail Ghulam Mohyuddin, Khalid Perveiz, Baklouti Syrine, Du, Qizhen, (2020). Application of machine learning tool to predict the porosity of clastic depositional system, Indus Basin, Pakistan 1. pp https://doi.org/10.1016/j.petrol.2020.107975.

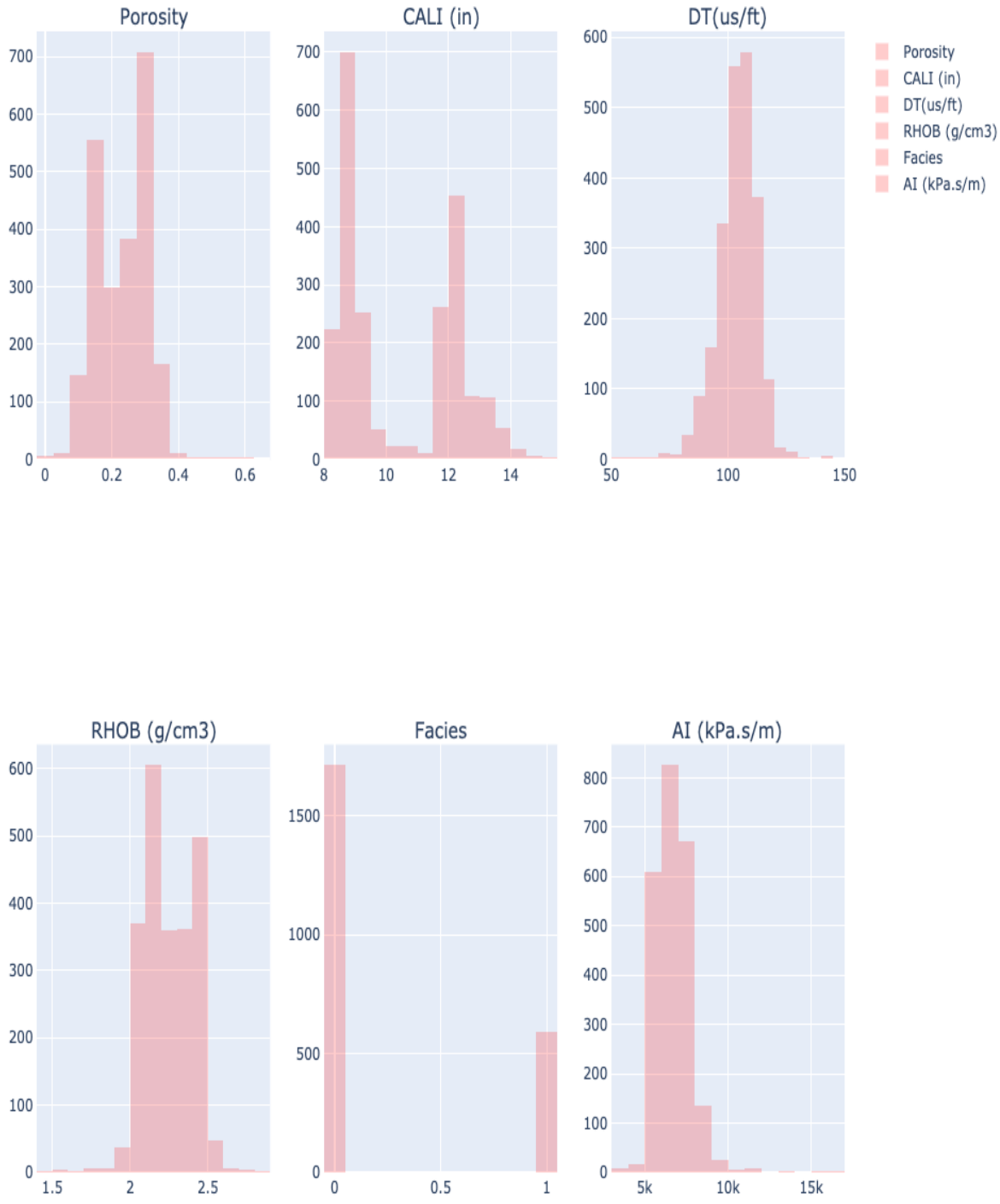
Ziegler, P. A 1981. Evolution of sedimentary basins in North West Europe. Habitat of Hydrocarbons on the Norwegian continental shelf (1986) pp 181-197.

Zhang L.H, Zhou C.C, Liu G.Q, et al. (2007). Origin and property differences of various types of low-porosity and low-permeability reservoirs and well logging evaluation strategies Petrol. Explore. Dev. 34 (6) pp. 702-710, 10.3321/j.issn:1000-0747.2007.06.012.

Zhong Z, Carr T.R., (2019). Application of a new hybrid particle swarm optimization-mixed kernels function-based support vector machine model for reservoir porosity prediction: a case study in Jacksonburg-

Stringtown oil field West Virginia USA Interpretation, 7 (1) 10.1190/INT-2018-0093.1. pp. 97-112.

Appendix 1: Histogram of the Well log features



Appendix 2: Facies classification in Python

Facies Classification

In [6]:
```
df['Facies']=np.zeros(len(df))
df['Facies']= np.where(df['GR (gAPI)']>=70, 1, 0)
combined_df['Facies']=np.zeros(len(combined_df))
combined_df['Facies']= np.where(combined_df['GR (gAPI)']>=70, 1, 0)
test['Facies']=np.zeros(len(test))
test['Facies']= np.where(test['GR (gAPI)']>=70, 1, 0)
```

In [7]:
```
combined_df
```

Out [7]:

		X (m)	Y (m)	Z (m)	MD (m)	CALI (in)	GR (gAPI)	DT (us / ft)	RHOB (g / cm3)	RT (ohm.m)	Porosity (m3 / m3)	AI (kPa.s / m)	Facies
33_12_1	0	437055.690000	6.786258e + 06	-2382.824000	2409,824	12.15	25.62	118.70	2.1145	126,1593	0.2950	5429.5449	0
	1	437055.690000	6.786258e + 06	-2478.824000	2505,824	12.15	23.03	118.42	2.0503	176.0741	0.3348	5277.1660	0
	2	437055.690000	6.786258e + 06	-2477,824,000	2504,824	12.17	28.16	117.01	2.0440	1516.6244	0.3361	5324.5840	0
	3	437055.690000	6.786258e + 06	-2476.824000	2503,824	12.18	28.30	113.42	2.0982	198.7117	0.3076	5638.7163	0
	4	437055.690000	6.786258e + 06	-2475.824000	2502,824	12.08	35.10	111.89	2.1022	129.6825	0.3052	5726.7749	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
33_9A_41	236	437771.591308	6.789415e + 06	-2555.400803	3942,141	8.46	56.55	101.35	2.2810	9.2014	0.2224	6859.9409	0
	237	437771.537349	6.789414e + 06	-2556.079173	3943,141	8.49	59.22	99.19	2.2712	7.7348	0.2277	6979.4390	0
	238	437771.483391	6.789413e + 06	-2556.757542	3944,141	8.62	56.53	97.11	2.3459	5.6340	0.1869	7363.3994	0
	239	437771.430167	6.789413e + 06	-2557.435783	3945.141	8.86	64.89	97.85	2.3909	4.8236	0.1624	7447.7900	0
	240	437771.381464	6.789412e + 06	-2558.113231	3946,141	9.10	43.77	96.43	2.3041	7.9635	0.2099	7284.3701	0

Appendix 3: Normalization in Python

Normalize the training and test dataset

In []:
```
#X = df [['AI', 'GR (gAPI)', 'X (m)', 'Y (m)', 'Z (m)']]
X  =  df [[ 'X (m)' , 'Y (m)' , 'Z (m)' , 'MD (m)' , 'CALI (in)' , 'DT (us / ft)' , ' AI (kPa.s / m) ' , ' RHOB (g / cm
Y  =  df [ 'Porosity (m3 / m3)' ]. values
```

In []:
```
#X_test = test ['Resampled AI (kPa.s / m)']. Values
#X_test = test [['AI', 'GR (gAPI)', 'X (m)', 'Y (m)', 'Z (m)']]
X_test  =  test [[ 'X (m)' , 'Y (m)' , 'Z (m)' , 'MD (m)' , 'CALI (in)' , 'DT (us / ft)' , ' AI (kPa.s / m) ' , ' RHOB
Y_test  =  test [ 'Porosity (m3 / m3)' ]. values
```

In []:
```
normalizer  =  preprocessing . StandardScaler (). fit ( pd . concat ([ X , X_test ]))
X  =  normalizer . transform ( X )
```

In []:
```
X_test  =  normalizer . transform ( X_test )
```

Appendix 4: Porosity prediction using Neural Network (NN) in Python

1. Neural Network

In []:
```
model = tf . keras . Sequential ()
model . add ( tf . keras . layers . Dense ( 5 , input_dim = 11 , activation = 'relu' ))
model . add ( tf . keras . layers . Dense ( 20 , activation = 'relu' ))
# model.add (tf.keras.layers.Dense (50, activation = 'relu'))
# model.add (tf.keras.layers.Dense (100, activation = 'relu'))
# model.add (tf.keras.layers.Dense (100, activation = 'relu'))
# model.add (tf.keras.layers.Dense (50, activation = 'relu'))
# model.add (tf.keras.layers.Dense (20, activation = 'relu'))
model . add ( tf . keras . layers . Dense ( 10 , activation = 'relu' ))
model . add ( tf . keras . layers . Dense ( 5 , activation = 'relu' ))
model . add ( tf . keras . layers . Dense ( 1 , activation = 'linear' ))
model . compile ( optimizer = 'adam' , loss = 'mean_squared_error' , metrics = [ 'accuracy' ])
```

In []:
```
history  =  model . fit ( X , Y , epochs = 100 , verbose = 1 , validation_split = 0.2 )
```

In []:
```
Y_predictions  =  model . predict ( X_test )
field_predictions  =  model . predict ( field_test )
```

In []:
```
fig  =  go . Figure ()
fig . add_trace ( go . Scatter (
    y = history . history [ 'loss' ],
    name = 'Training loss (mse)' ,
    mode  =  'lines' ,
    marker_color = 'rgba (0,0, 255, .4)'
))
fig . add_trace ( go . Scatter (
    y = history . history [ 'val_loss' ],
    name = 'Validation loss (mse)' ,
    mode = 'lines' ,
    marker_color = 'rgba (255, 0, 0, .4)' ,
))
And
And
And
# Set options common to all traces with fig.update_traces
fig . update_traces ( marker_line_width = 2 , marker_size = 8 ,)
fig . update_layout ( title = 'Loss function of Neural Network' ,
                width = 1000 , height = 600 ,
                )
fig . update_xaxes ( title_text = "Epochs" )
fig . update_yaxes ( title_text = 'loss (mse)' )
And
fig . show ()
```

In []:
```
fig  =  go . Figure ()
fig . add_trace ( go . Scatter (
     y = Y_test ,
    name = 'Actual values' ,
    mode  =  'lines' ,
    marker_color = 'rgba (0,0, 255, .4)'
))
fig . add_trace ( go . Scatter (
     y = Y_predictions [:, 0 ],
    name = 'Predicted values' ,
    mode = 'lines' ,
    marker_color = 'rgba (255, 0, 0, .4)' ,
))
And
And
And
# Set options common to all traces with fig.update_traces
fig . update_traces ( marker_line_width = 2 , marker_size = 8 ,)
fig . update_layout ( title = 'Predicted vs actual values for well M' ,
                 width = 1000 , height = 600 ,
                 )
# fig.update_xaxes (title_text = "Acoustic Impedance (Normalized)")
fig . update_yaxes ( title_text = 'Porosity (Normalized)' )
And
fig . show ()
```

In []:
```
fig  =  go . Figure ()
fig . add_trace ( go . Scatter (
     x = Y_test , y  =  normalizer . inverse_transform ( X_test ) [:, 3 ],
    name = 'Actual values' ,
    mode  =  'lines' ,
    marker_color = 'rgba (0,255, 0, .4)'
))
fig . add_trace ( go . Scatter (
     x = Y_predictions [:, 0 ], y  =  normalizer . inverse_transform ( X_test ) [:, 3 ],
    name = 'Predicted values' ,
    mode = 'lines' ,
    marker_color = 'rgba (255, 0, 0, .4)' ,
))
And
And
And
# Set options common to all traces with fig.update_traces
fig . update_traces ( marker_line_width = 1 , marker_size = 5 , line_width  =  1 )
fig . update_layout ( title = 'Predicted vs actual values for well M' ,
                 width = 500 , height = 700 ,
                 )
fig . update_xaxes ( title_text = "Porosity" )
fig . update_yaxes ( title_text = 'Depth (MD)' )
And
fig . show ()
```

In []:
```
#Well prediction
And
metrics ( Y_test , Y_predictions )
```

Appendix 5: Porosity prediction using support vector machine (SVM) in Python

In []:
```
regr  =  svm . SVR ()
```

In []:
```
regr . fit ( X , Y )
```

In []:
```
svm_prediction  =  regr . predict ( X_test )
svm_field_prediction  =  regr . predict ( field_test )
```

In []:
```
fig  =  go . Figure ()
fig . add_trace ( go . Scatter (
     y = Y_test ,
    name = 'Actual values' ,
    mode  =  'lines' ,
    marker_color = 'rgba (0,0, 255, .4)'
))
fig . add_trace ( go . Scatter (
     y = svm_prediction ,
    name = 'Predicted values' ,
    mode = 'lines' ,
    marker_color = 'rgba (255, 0, 0, .4)' ,
))
And
And
And
# Set options common to all traces with fig.update_traces
fig . update_traces ( marker_line_width = 2 , marker_size = 8 ,)
fig . update_layout ( title = 'Predicted vs actual values for well M' ,
                  width = 1000 , height = 600 ,
                  )
# fig.update_xaxes (title_text = "Acoustic Impedance (Normalized)")
fig . update_yaxes ( title_text = 'Porosity (Normalized)' )
And
fig . show ()
```

In []:
```
fig  =  go . Figure ()
fig . add_trace ( go . Scatter (
     x = Y_test , y  =  normalizer . inverse_transform ( X_test ) [:, 3 ],
    name = 'Actual values' ,
    mode  =  'lines' ,
    marker_color = 'rgba (0,255, 0, .4)'
))
fig . add_trace ( go . Scatter (
     x = svm_prediction , y  =  normalizer . inverse_transform ( X_test ) [:, 3 ],
    name = 'Predicted values' ,
    mode = 'lines' ,
    marker_color = 'rgba (255, 0, 0, .4)' ,
))
And
And
And
# Set options common to all traces with fig.update_traces
fig . update_traces ( marker_line_width = 1 , marker_size = 5 , line_width  =  1 )
fig . update_layout ( title = 'Predicted vs actual values for well M' ,
                  width = 500 , height = 700 ,
                  )
fig . update_xaxes ( title_text = "Porosity" )
fig . update_yaxes ( title_text = 'Depth (MD)' )
And
fig . show ()
```

In []:
```
#for well
metrics ( Y_test , svm_prediction )
```

Appendix 6: Porosity prediction using Decision Tree (DT) in Python

3. Decision tree

In []:
```
from  sklearn  import  tree
And
clf  =  tree . DecisionTreeRegressor ()
clf  =  clf . fit ( X , Y )
tree_prediction  =  clf . predict ( X_test )
tree_field_prediction  =  clf . predict ( field_test )
```

In []:
```
fig  =  go . Figure ()
fig . add_trace ( go . Scatter (
     y = Y_test ,
    name = 'Actual values' ,
    mode  =  'lines' ,
    marker_color = 'rgba (0,0, 255, .4)'
))
fig . add_trace ( go . Scatter (
     y = tree_prediction ,
    name = 'Predicted values' ,
    mode = 'lines' ,
    marker_color = 'rgba (255, 0, 0, .4)' ,
))
And
And
And
# Set options common to all traces with fig.update_traces
fig . update_traces ( marker_line_width = 2 , marker_size = 8 ,)
fig . update_layout ( title = 'Predicted vs actual values for well M' ,
                  width = 1000 , height = 600 ,
                  )
# fig.update_xaxes (title_text = "Acoustic Impedance (Normalized)")
fig . update_yaxes ( title_text = 'Porosity (Normalized)' )
And
fig . show ()
```

In []:
```
fig  =  go . Figure ()
fig . add_trace ( go . Scatter (
     x = Y_test , y  =  normalizer . inverse_transform ( X_test ) [:, 3 ],
    name = 'Actual values' ,
    mode  =  'lines' ,
    marker_color = 'rgba (0,255, 0, .4)'
))
fig . add_trace ( go . Scatter (
     x = tree_prediction , y  =  normalizer . inverse_transform ( X_test ) [:, 3 ],
    name = 'Predicted values' ,
    mode = 'lines' ,
    marker_color = 'rgba (255, 0, 0, .4)' ,
))
And
And
And
# Set options common to all traces with fig.update_traces
fig . update_traces ( marker_line_width = 1 , marker_size = 5 , line_width  =  1 )
fig . update_layout ( title = 'Predicted vs actual values for well M' ,
                  width = 500 , height = 700 ,
                  )
fig . update_xaxes ( title_text = "Porosity" )
fig . update_yaxes ( title_text = 'Depth (MD)' )
And
fig . show ()
```

In []:
```
#For well
And
metrics ( Y_test , tree_prediction )
```

Appendix 7: Porosity prediction using K Nearest Neighbour network (KNN) in Python

K Nearest Neighbor

In []:
```
from sklearn import neighbors
```

In []:
```
knn  =  neighbors . KNeighborsRegressor ( 5 , weights = 'distance' )
knn_prediction  =  knn . fit ( X , Y ). predict ( X_test )
knn_field_prediction  =  knn . predict ( field_test )
```

In []:
```
fig  =  go . Figure ()
fig . add_trace ( go . Scatter (
     y = Y_test ,
    name = 'Actual values' ,
    mode  =  'lines' ,
    marker_color = 'rgba (0,0, 255, .4)'
))
fig . add_trace ( go . Scatter (
     y = knn_prediction ,
    name = 'Predicted values' ,
    mode = 'lines' ,
    marker_color = 'rgba (255, 0, 0, .4)' ,
))
And
And
And
# Set options common to all traces with fig.update_traces
fig . update_traces ( marker_line_width = 2 , marker_size = 8 ,)
fig . update_layout ( title = 'Predicted vs actual values for well M' ,
                 width = 1000 , height = 600 ,
                 )
# fig.update_xaxes (title_text = "Acoustic Impedance (Normalized)")
fig . update_yaxes ( title_text = 'Porosity (Normalized)' )
And
fig . show ()
```

In []:
```
fig  =  go . Figure ()
fig . add_trace ( go . Scatter (
     x = Y_test , y  =  normalizer . inverse_transform ( X_test ) [:, 3 ],
    name = 'Actual values' ,
    mode  =  'lines' ,
    marker_color = 'rgba (0,255, 0, .4)'
))
fig . add_trace ( go . Scatter (
     x = knn_prediction , y  =  normalizer . inverse_transform ( X_test ) [:, 3 ],
    name = 'Predicted values' ,
    mode = 'lines' ,
    marker_color = 'rgba (255, 0, 0, .4)' ,
))
And
And
And
# Set options common to all traces with fig.update_traces
fig . update_traces ( marker_line_width = 1 , marker_size = 5 , line_width  =  1 )
fig . update_layout ( title = 'Predicted vs actual values for well M' ,
                width = 500 , height = 700 ,
                )
fig . update_xaxes ( title_text = "Porosity" )
fig . update_yaxes ( title_text = 'Depth (MD)' )
And
fig . show ()
```

In []:
```
metrics ( Y_test , knn_prediction )
```

Appendix 8: Porosity prediction using Bayesian Ridge Regression (BRR) in Python

5 Bayesian

In []:
```
from  sklearn . linear_model  import  BayesianRidge , LinearRegression
```

In []:
```
clf  =  BayesianRidge ( compute_score = True )
clf . fit ( X , Y )
bayesian_prediction  =  clf . predict ( X_test )
bayesian_field_prediction  =  clf . predict ( field_test )
```

In []:
```
fig  =  go . Figure ()
fig . add_trace ( go . Scatter (
     y = Y_test ,
    name = 'Actual values' ,
    mode  =  'lines' ,
    marker_color = 'rgba (0,0, 255, .4)'
))
fig . add_trace ( go . Scatter (
     y = bayesian_prediction ,
    name = 'Predicted values' ,
    mode = 'lines' ,
    marker_color = 'rgba (255, 0, 0, .4)' ,
))
And
And
And
# Set options common to all traces with fig.update_traces
fig . update_traces ( marker_line_width = 2 , marker_size = 8 ,)
fig . update_layout ( title = 'Predicted vs actual values for well M' ,
                  width = 1000 , height = 600 ,
                  )
# fig.update_xaxes (title_text = "Acoustic Impedance (Normalized)")
fig . update_yaxes ( title_text = 'Porosity (Normalized)' )
And
fig . show ()
```

In []:
```
fig  =  go . Figure ()
fig . add_trace ( go . Scatter (
     x = Y_test , y  =  normalizer . inverse_transform ( X_test ) [:, 3 ],
    name = 'Actual values' ,
    mode  =  'lines' ,
    marker_color = 'rgba (0,255, 0, .4)'
))
fig . add_trace ( go . Scatter (
     x = bayesian_prediction , y  =  normalizer . inverse_transform ( X_test ) [:, 3 ],
    name = 'Predicted values' ,
    mode = 'lines' ,
    marker_color = 'rgba (255, 0, 0, .4)' ,
))
And
And
And
# Set options common to all traces with fig.update_traces
fig . update_traces ( marker_line_width = 1 , marker_size = 5 , line_width  =  1 )
fig . update_layout ( title = 'Predicted vs actual values for well M' ,
                  width = 500 , height = 700 ,
                  )
fig . update_xaxes ( title_text = "Porosity" )
fig . update_yaxes ( title_text = 'Depth (MD)' )
And
fig . show ()
```

In []:
```
metrics ( Y_test , bayesian_prediction )
```

Appendix 9: Porosity prediction using Linear Regression (LR) in Python

6. Linear Regression

In []:
```
ols  =  LinearRegression ()
ols . fit ( X , Y )
linear_prediction  =  ols . predict ( X_test )
linear_field_prediction  =  ols . predict ( field_test )
```

In []:
```
fig  =  go . Figure ()
fig . add_trace ( go . Scatter (
     y = Y_test ,
    name = 'Actual values' ,
    mode  =  'lines' ,
    marker_color = 'rgba (0,0, 255, .4)'
))
fig . add_trace ( go . Scatter (
     y = linear_prediction ,
    name = 'Predicted values' ,
    mode = 'lines' ,
    marker_color = 'rgba (255, 0, 0, .4)' ,
))
And
And
And
# Set options common to all traces with fig.update_traces
fig . update_traces ( marker_line_width =2 , marker_size = 8 ,)
fig .update_layout ( title = 'Predicted vs actual values for well M' ,
                  width =1000 , height = 600 ,
                  )
# fig.update_xaxes (title_text = "Acoustic Impedance (Normalized)")
fig . update_yaxes (title_text = 'Porosity (Normalized)' )
And
fig . show ()
```

In []:
```
fig  =  go .Figure ()
fig . add_trace ( go .Scatter (
     x = Y_test , y  =  normalizer . inverse_transform ( X_test ) [:,3 ],
    name ='Actual values' ,
    mode  = 'lines' ,
    marker_color= 'rgba (0,255, 0, .4)'
))
fig . add_trace ( go . Scatter (
     x = linear_prediction , y  =  normalizer . inverse_transform ( X_test ) [:, 3 ],
    name = 'Predicted values' ,
    mode = 'lines' ,
    marker_color = 'rgba (255, 0, 0, .4)' ,
))
And
And
And
# Set options common to all traces with fig.update_traces
fig . update_traces ( marker_line_width = 1 , marker_size = 5 , line_width  =  1 )
fig . update_layout ( title = 'Predicted vs actual values for well M' ,
                 width = 500 , height = 700 ,
                 )
fig . update_xaxes ( title_text = "Porosity" )
fig . update_yaxes ( title_text = 'Depth (MD)' )
And
fig . show ()
```

In []:
```
metrics ( Y_test , linear_prediction )
```

Appendix 10: Porosity prediction using Random Forest (RF) in Python

7. Random forest

In []:
```
from  sklearn . ensemble  import  RandomForestRegressor
```

In []:
```
regr  =  RandomForestRegressor ( max_depth = 5 , random_state = 0 )
regr . fit ( X , Y )
forest_prediction  =  regr . predict ( X_test )
forest_field_prediction  =  regr . predict ( field_test )
```

In []:
```
fig  =  go . Figure ()
fig . add_trace ( go . Scatter (
     y = Y_test ,
    name = 'Actual values' ,
    mode  =  'lines' ,
    marker_color = 'rgba (0,0, 255, .4)'
))
fig . add_trace ( go . Scatter (
     y = forest_prediction ,
    name = 'Predicted values' ,
    mode = 'lines' ,
    marker_color = 'rgba (255, 0, 0, .4)' ,
))
And
And
And
# Set options common to all traces with fig.update_traces
fig . update_traces ( marker_line_width = 2 , marker_size = 8 ,)
fig . update_layout ( title = 'Predicted vs actual values for well M' ,
                  width = 1000 , height = 600 ,
                  )
# fig.update_xaxes (title_text = "Acoustic Impedance (Normalized)")
fig . update_yaxes ( title_text = 'Porosity (Normalized)' )
And
fig . show ()
```

In []:
```
fig  =  go . Figure ()
fig . add_trace ( go . Scatter (
     x = Y_test , y  =  normalizer . inverse_transform ( X_test ) [:, 3 ],
    name = 'Actual values' ,
    mode  =  'lines' ,
    marker_color = 'rgba (0,255, 0, .4)'
))
fig . add_trace ( go . Scatter (
     x = forest_prediction , y  =  normalizer . inverse_transform ( X_test ) [:, 3 ],
    name = 'Predicted values' ,
    mode = 'lines' ,
    marker_color = 'rgba (255, 0, 0, .4)' ,
))
And
And
And
# Set options common to all traces with fig.update_traces
fig . update_traces ( marker_line_width = 1 , marker_size = 5 , line_width  =  1 )
fig . update_layout ( title = 'Predicted vs actual values for well M' ,
                  width = 500 , height = 700 ,
                  )
fig . update_xaxes ( title_text = "Porosity" )
fig . update_yaxes ( title_text = 'Depth (MD)' )
And
fig . show ()
```

In []:
```
metrics ( Y_test , forest_prediction )
```