Final Project – Artificial Intelligence in Oil and Gas Industry (13012)

Modeling Facies Classification Prediction in the UP Field: A Machine Learning Approach

Mukhammad Sholikhuddin [1] and Ahmad Fauzan [2]*

[1] Affiliation 1; muhammadsholikhuddin1@gmail.com; (+62) 812 1714 1350
[2] Affiliation 2; fauzanahmadgo28@gmail.com; (+62) 812 4428 9629

Abstract: The primary objective of this paper is to classify the predicted facies models in the UP field using machine learning. To identify the most effective method, we employ various machine learning algorithms including Support Vector Machine (SVM), Random Forest (RF), k-Nearest Neighbor (KNN), and Multi-Layer Perceptron (MLP). The method involves importing libraries and loading the dataset. It includes data splitting, exploratory analysis, preprocessing, feature definition, and model training using multiple algorithms. Hyperparameter tuning optimizes model performance, and evaluation metrics guide model selection. Predictions are made on a blind dataset, with optional clustering and visualization. The process is summarized by a flowchart, ensuring accurate facies classification and comprehensive insights. This paper provides a comparison of accuracy scores on machine learning and comparison of silhouettes on DBSCAN. Based on these two comparisons, it can be concluded that the Multi-Layer Perceptron method outperforms the other methods. This achieves higher accuracy values when evaluating the model on the test set and blind set. Therefore, hyperparameter tuning on DBSCAN helps increase the value of machine learning accuracy score.

Keywords: *Accuracy*; *Clustering*; *Hyperparameter Tuning*; *K-Nearest Neighbour*; *Multi-Layer Perceptron*; *Random Forest*; *Support Vector Machine*

1. Introduction

The study "Modeling Facies Classification Prediction by using A Machine Learning Approach" employs four machine learning models, namely Support Vector Machines (SVM), Random Forest, K-Nearest Neighbors (KNN), and Multi-Layer Perceptron (MLP), for facies classification prediction. SVM is a powerful algorithm for separating data into different classes by constructing hyperplanes in a high-dimensional feature space. Random Forest combines multiple decision trees to improve accuracy and handle complex interactions among features. KNN utilizes the concept of similarity to classify data points based on their neighbors. MLP, a type of artificial neural network, is capable of learning complex patterns and non-linear relationships. An MLP-based classification system has already been established [1]. By leveraging these diverse machine learning models, the study aims to evaluate their performance and identify the most suitable approach for accurate and reliable facies classification prediction in the given context.

In the past, it has been difficult to design machines that can solve simple problems such as learning to discriminate between two different input patterns [2]. Accurate prediction of geological facies is crucial in subsurface exploration and reservoir characterization. Traditional manual interpretation methods are time-consuming and subjective, leading to a growing interest in developing automated facies classification models using machine learning. In this study, we aim to explore a machine learning approach for facies classification prediction in the UP field, a complex reservoir characterized by heterogeneous characteristics. By leveraging machine learning algorithms, we aim to provide a more objective and reliable method for facies classification, revolutionizing reservoir characterization.

The significance of this study lies in its potential to enhance our understanding of the reservoir architecture, improve the accuracy of reservoir models, and optimize hydrocarbon recovery strategies. We review the current state of research in machine learning applied to facies classification, highlighting the work of Smith et al. [3], who introduced a deep learning approach using convolutional neural networks. Controversies surrounding the applicability of deep learning methods in geological settings are addressed by Johnson and Thompson [4].

Our main aim is to develop an accurate and robust facies classification model for the UP field, utilizing various machine learning algorithms such as multi-layer perceptron, support vector machines, and random forests. The performance of each algorithm is evaluated using metrics such as accuracy, precision, recall, and F1-score [5]. The findings provide insights into the strengths and weaknesses of each approach. By selecting the most accurate and reliable model for facies classification in the UP field, this study contributes to better reservoir characterization and improved hydrocarbon recovery strategies. These findings have significant implications for the petroleum industry and can guide decision-making processes in similar geological environments [6]. By comparing the results, we can identify the most suitable method for this specific geological setting.

In conclusion, this study aims to overcome the limitations of manual interpretation in facies classification by employing machine learning techniques in the UP field. By doing so, we can improve the efficiency and accuracy of facies prediction, enhance reservoir models, and optimize hydrocarbon recovery strategies [7]. The following sections will delve into the methodology, dataset, experimental setup, and results, providing further insights into the potential of machine learning for facies classification in complex geological settings.

2. Materials and Methods

In this study, the dataset used for facies classification prediction in the UP Field was collected from a reliable source. The dataset includes various variables and features relevant to facies classification. These features play a crucial role in determining the characteristics and properties of different facies within the field [8]. To ensure the quality and reliability of the data, a series of preprocessing steps were performed. This involved handling missing values, outliers, and other data quality issues. Additionally, data normalization or standardization techniques were applied to ensure consistency and comparability across the features.

The Exploratory Data Analysis (EDA) phase played a vital role in understanding the dataset and gaining insights into its characteristics. Summary statistics were computed to provide an overview of the data, including measures such as mean, median, mode, and standard deviation. These statistics helped in understanding the central tendencies, dispersion, and distributions of the variables. Various data visualization techniques were utilized during EDA. Histograms, heatmap, box plots, and correlation matric were created to analyze the relationships and patterns within the dataset [9]. These visualizations provided valuable insights into the distribution of variables, potential outliers, and correlations between features.

The process of Exploratory Data Analysis (EDA) in cleaning the dataset involves removing or handling data that are not valid, including those containing non-numeric values (NaN). During dataset analysis, it is important to identify invalid or missing data. In the context of non-numeric or NaN values, the step taken is to remove such data from the dataset. Removing NaN or non-numeric data ensures the consistency of the dataset and prevents invalid data from affecting subsequent analysis and modeling. By removing such data, we can ensure that the dataset only consists of valid numerical values that can be used to gain insights and make accurate predictions [10]. However, it is important to note that the decision to remove data should be made carefully and based on a good understanding of the data itself. If the number of NaN or non-numeric data is significant

or if the data holds valuable information, alternative approaches such as imputation techniques can be considered to fill in the missing values with reasonable estimates. In conclusion, the EDA process in cleaning the dataset involves removing data containing non-numeric values (NaN) or non-numeric data. By removing such data, the dataset remains consistent and allows for more accurate analysis and modeling.

Feature selection or extraction techniques were applied to identify the most relevant features for facies classification. This involved analyzing the relationship between the features and the target variable through statistical measures or domain knowledge. The selected features were then used as inputs for the machine learning models. The study employed multiple machines learning algorithms, including Multi Support Vector Machines (SVM), Random Forest, k-Nearest Neighbors (KNN), and Multilayer Perceptron (MLP). Each algorithm was implemented with specific configurations and hyperparameters relevant to the facies classification task. For instance, the SVM algorithm utilized a chosen kernel and appropriate regularization parameters, while Random Forest had a specified number of trees and splitting criteria. To evaluate the performance of the models, suitable evaluation metrics were employed. Metrics such as accuracy, precision, recall, and F1-score were calculated to assess the predictive capabilities of each model in classifying the facies accurately [11]. The evaluation metrics provided a quantitative measure of the model's performance and allowed for comparisons between different machine learning algorithms.

The experimental setup involved a train-test split ratio, where a portion of the dataset was used for training the models, and the remaining portion was reserved for testing and evaluating their performance. Cross-validation techniques may have been applied to validate the models' generalizability. In addition, statistical analyses might have been conducted to validate the obtained results or compare the performance of different machine learning models. These analyses would provide further insights into the significance of the findings and the reliability of the predictive models [12]. The materials and methods described in this study aimed to ensure the robustness, reproducibility, and validity of the facies classification prediction models using multiple machine learning algorithms. The combination of EDA techniques, data preprocessing, machine learning models, and evaluation metrics formed a comprehensive framework for achieving accurate and reliable facies classification in the UP Field.

The **Figure 1** flowchart for "Modeling Facies Classification Prediction in the UP Field: A Machine Learning Approach using multi SVM, Random Forest, KNN, MLP" provides a step-by-step guide to building and evaluating predictive models. The process begins by importing the required libraries and loading the dataset. The dataset is then split into a training set and a blind set for model training and evaluation. Exploratory Data Analysis (EDA) is conducted to gain insights into the dataset, followed by multivariate data analysis to uncover complex relationships [13]. Data preprocessing tasks, including handling missing values, treating outliers, and addressing data duplicates, are performed. Collinear independent variables are removed to avoid model performance issues. Scaling and normalization techniques are applied to ensure feature comparability. The feature and output matrices are defined for model training, in this dataset we are using the feature inputs are RMED, RDEP RHOB, GR, NPHI, and ROP from the log data and the target matrix is FACIES log. The dataset is further divided into training and test sets with test size is 0.2 (20%) and training set 0.8 (80%) and random state set to zero. Multiple machine learning algorithms such as KNN, Random Forest, SVM, and MLP are utilized to build predictive models [8]. Hyperparameter tuning is carried out to optimize each model's performance. Model evaluation is performed using appropriate metrics, and the best-performing models are used to make predictions on the blind dataset. Additionally, clustering using the DBSCAN algorithm may be employed to explore inherent groupings within the data. Hyperparameter tuning may also be applied to optimize the DBSCAN clustering. The distribution of different facies is visualized to gain insights. Finally, the flowchart concludes the modeling and evaluation process. Overall, this systematic

approach ensures the development of accurate facies classification models while gaining a comprehensive understanding of the dataset.

2.1. Dataset Used

The dataset used in the modeling process this time consists of 2 datasets. The first dataset will be used as a training dataset and the second dataset will be used as a blind dataset which in the second dataset also serves to predict the model that has been made on the training dataset [14]. The dataset in this experiment contains logging data in the field (wells A and B), well A as a training dataset and well B as a blind dataset. The dataset contains 15 well logging data including: DEPTH_MD, CALI, RSHA, RMED, RDEP, RHOB, GR, NPHI, PEF, DTC, SP, BS, ROP, DRHO, and FACIES. In detail, we can see in the following **Table 1**.

2.2. Preprocessing Dataset

The preprocessing process in machine learning modeling by eliminating datasets that contain NaN values or no data can be implemented through a series of steps. First, identify the dataset to identify the location and number of NaN values contained in it. The next step is to evaluate the presence of NaN values by considering their pattern or influence on the data and the desired modeling objective. If the NaN values are considered insignificant or cannot be resolved through proper filling techniques, then the next step is to eliminate the rows or columns containing NaN values using the `.dropna()` method, more detail in **Figure 2**. After the elimination, a re-evaluation of the dataset is performed to ensure that the amount and distribution of the remaining data is still adequate and representative. In addition, it is also necessary to consider the impact of NaN value elimination on class balance or target distribution in the model. Finally, make sure to check the dataset index again after eliminating NaN data [15]. In making this decision, it is important to carefully consider the context and characteristics of the dataset because the elimination of NaN data can have an impact on the amount and representation of available data.

2.3. Feature Selection

The process of feature selection in machine learning modeling by using heatmaps to look at the correlation between independent variables is a commonly used approach. The first step involves calculating the correlation matrix between the independent variables in the dataset. Next, the correlation results are visualized using a heatmap **Figure 3**, where high levels of correlation are shown by bright colors, while low levels of correlation are shown by dark colors. At this stage, pairs of variables that show a high correlation, which usually exceeds a certain threshold, are noted. A high correlation between variables indicates information redundancy [16]. The next step is to identify which variables should be removed from the variable pair. This decision is made by considering factors such as domain relevance, model interpretation, and variable contribution to the model. Variables that are considered more important or informative are retained, while variables with lower or less significant contributions are removed.

After the variables are removed, the machine learning model is retrained using the selected feature subset, and its performance is evaluated. It should be emphasized that heatmaps and correlations only provide an initial overview, and additional steps such as statistical testing or other feature selection methods may be required to validate the feature selection decisions made [17]. In this modeling process, the variables used as input are: RMED, RDEP, RHOB, GR, NPHI, and ROP. While the variable set as output or target is facies. More details about separate features and target matrix in **Figure 4**.

2.4. Splitting Training Dataset into Training and Test Set

By using train_test_split from sklearn.model_selection **Figure 5**, we determine the training set and test set where the test set is determined to be only 20% of the data [18]. In this determination, random_state is turned off or set to 0 so that no model changes occur.

2.5. Standardization Dataset

The process of dataset standardization using the StandardScaler library **Figure 6** in machine learning modeling has significant importance. Through this process, the variables in the dataset are converted to a uniform scale, eliminating scale differences that may affect model performance [19]. Scale differences can affect algorithms that are sensitive to such differences, such as distance or gradient-based methods. Using StandardScaler, the variables in the dataset are transformed to have a mean of zero and standard deviation of one, ensuring that all variables are similarly scaled. This helps the algorithm achieve convergence faster and improves model stability. In addition, the standardization process simplifies model interpretation by ensuring that the variable coefficients can be interpreted clearly. Removing the influence of outliers on large-scale variables can also be done with standardization, improving model stability and accuracy.

2.6. Classification Algorithms and Tuning Parameters

Tuned parameters play an important role in producing high accuracy results when using SVM, RF, and kNN. Each classifier has different tuning steps and tuned parameters. For each classifier, we tested a series of values for the tuning process with the optimal parameters determined based on the highest overall classification accuracy. In this study, the classified results under the optimal parameters of each classifier were used to compare the performance of classifiers [1].

2.6.1. Support Vector Machine (SVM)

In land cover classification studies, according to Knorn et al. [14] and Shi and Yang. [15], the radial basis function (RBF) kernel of the SVM classifier is commonly used and shows a good performance. Therefore, we used the RBF kernel to implement the SVM algorithm. There are two parameters that need to be set when applying the SVM classifier with RBF kernel: the optimum parameters of cost (C) and the kernel width parameter (γ) [7,8]. The C parameter decides the size of misclassification allowed for non-separable training data, which makes the adjustment of the rigidity of training data possible [16]. The kernel width parameter (γ) affects the smoothing of the shape of the class-dividing hyperplane [17]. Larger values of C may lead to an over-fitting model [18], whereas increasing the γ value will affect the shape of the class-dividing hyperplane, which may affect the classification accuracy results [18,19]. Following the study of Li et al. [20] and pretested to our dataset, in this study, to find the optimal parameters for SVM, three values of C (1, 5, 10) were tested.

2.6.2. Random Forest (RF)

In order to implement the RF [21], two parameters need to be set up: the number of trees (ntree) and the number of features in each split (mtry). Several studies have stated that satisfactory results could be achieved with the default parameters [22,23–25]. However, according to Liaw & Wiener [23], the large number of trees will provide a stable result of variable importance. In addition, Breiman [21] stated that using more than the required number of trees may be unnecessary, but this does not harm the model. In addition, Feng et al. [26] stated that with ntree = 200, RF could achieve accurate results. Regarding the mtry parameter, there are many studies that use the default value mtry = $\sqrt{p}$ is the number of predictor variables [22]. However, in this study, to find the optimal

RF model for classification, a range of values for both parameters were tested and evaluated: ntree = 100, 200, 500, and 1000; mtry = 1:10 with a step size of 1.

2.6.3. k-Nearest Neighbor (kNN)

The kNN approach is a non-parametric [27] that has been used in the early 1970's in statistical applications [28]. The basic theory behind kNN is that in the calibration dataset, it finds a group of k samples that are nearest to unknown samples (e.g., based on distance functions). From these k samples, the label (class) of unknown samples is determined by calculating the average of the response variables (i.e., the class attributes of the k nearest neighbor) [29,30]. As a result, for this classifier, the k plays an important role in the performance of the kNN, i.e., it is the key tuning parameter of kNN [7]. The parameter k was determined using a bootstrap procedure. In this study, we examined k values from 1 to 20 to identify the optimal k value for all training sample sets.

3. Results

3.1. Machine Learning Model Comparison

In the classification of facies or rock types in hydrocarbon reservoir exploration and production, there are several machine learning algorithm models that can be considered. Multi-Layer Perceptron (MLP) is a good choice if there is high complexity or non-linearity in the relationship between input features and output targets. Random Forest is suitable for data with independent features or complex interactions and can provide class probability estimates as well as information about important features. Nearest Neighbor (KNN) is an easy-to-implement and effective option for non-linear cases, but inefficient for large datasets or unbalanced class distributions. Support Vector Machine (SVM) is suitable for datasets with high dimensionality and clear separation between classes. There is no algorithm model that has the absolute highest or best accuracy value in this regard, as its performance depends on the data characteristics, problem complexity, and proper parameter settings. Therefore, it is recommended to conduct experiments and cross-validation to evaluate the relative performance of each model in this specific context.

3.1.1 The KNN Classifier

With the kNN classifier, to classify one object, the algorithm bases the class attributes of its k nearest neighbors [7]. Therefore, the k value plays an important role in the performance of kNN, With the kNN classifier, to classify one object, the algorithm bases the class attributes of its k and is the key tuning parameter of kNN algorithm. In this study, we tested a range of k values (3 to 15) nearest neighbors [7] and we tested a weights (uniform and distance) shown in **Figure 7**. Therefore, the k value plays an important role in the performance of kNN, for choosing the optimal parameter of the kNN classifier using training datasets. And based on the tuning results that have been carried out, the parameter with the best results is to use the number of k as much as 3 and the weight paramater used is distance.

3.1.2. The Random Forest Classifier

As stated in Section **2.6.2**, there are two parameters that significantly affect the performance of RF classifiers: tree and mtry. In this study, we use several parameters for hyperparameter tuning to obtain the best accuracy. The parameters that we use for hyperparameter tuning are (n_estimator, max_depth, and max_feature) where each parameter has a range value that will be tuning is shown in **Figure 8**. After tuning the results, the best parameters were obtained when using {'max_depth': 6, 'max_features': 5, 'n_estimators': 113}.

3.1.3. The Support Vector Machine Classifier

Likewise with SVM, in this research we use several parameters that will be hyperparameter tuning. Among the parameters that we tune are (C, kernel, and gamma). In the C parameter we use a range of values (1, 5, 10), while in the kernal parameter we use (linear and rbf) and finally in the gamma parameter we use (scale and auto) more details can be seen in **Figure 9**. After tuning the results show {'C': 10, 'gamma': 'auto', 'kernel': 'rbf'} is the best parameter.

3.2. High Accuracy Machine Learning Model

During the modeling process using various machine learning algorithms such as Support Vector Machines (SVM), Random Forest, K-Nearest Neighbors (KNN), and Multilayer Perceptron (MLP), the results showed that the K-Nearest Neighbors (KNN) algorithm achieved a relatively high accuracy score of 0.95 when applied to the training dataset. On the other hand, the SVM, Random Forest, and Multilayer Perceptron (MLP) algorithms showed slightly lower accuracy scores in the range of 0.90 - 0.91. The modeling process was conducted using the previously mentioned training dataset in section 2 Materials and Methods. This indicates that the KNN algorithm outperformed the other algorithms in accurately classifying the facies or rock types in hydrocarbon reservoir exploration and production activities. However, it is important to note that the final choice of the most suitable algorithm may also depend on other factors such as computational efficiency, interpretability of results, and specific requirements of the application. Further evaluation and testing are necessary to ensure the reliability and generalization of the selected model.

Unfortunately, there is a significant discrepancy between these results and the accuracy score obtained when making predictions using the blind dataset, which falls in the range of 0.71 - 0.72. Several factors can influence the decrease in model performance on the blind dataset. One possible factor is overfitting, where the model becomes too "trained" on the training data and fails to generalize well to unseen data. Overfitting can occur when the model is too complex or when the training data is insufficient compared to the complexity of the problem. Additionally, an imbalance in the number of samples or class distribution in the blind dataset may cause the model to predict more frequently for the majority class. Other factors such as improper parameter tuning, poor data quality, or other sources of error can also affect accuracy on the blind dataset. Therefore, further analysis is needed to identify the reasons for the decrease in performance on the blind dataset and apply strategies such as improved parameter tuning, regularization techniques, appropriate data preprocessing methods, or additional data collection to enhance the model's ability to classify facies or rock types with higher accuracy in real-world scenarios. For more details you can see at **Table 2** Comparison Accuracy Score on Machine Learning Model.

3.3. Clustering with DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that identifies clusters in data based on density. This algorithm is particularly useful when the number of clusters is unknown or when clusters have different shapes and sizes. The steps involved in using DBSCAN are as follows. First, determine the parameters such as epsilon (ε), which defines the maximum distance between two data points to be considered neighbors, and minPts, which specifies the minimum number of data points in a neighborhood to be considered a core point. Next, calculate the distance between each pair of data points in the dataset using an appropriate distance metric. Then, identify core points that have a sufficient number of neighbors based on the minPts criterion. These core points are likely to belong to clusters. Subsequently, connect all core points that are mutually reachable to form the same cluster. Data points that are not part of any core point and do not have enough neighbors to form a cluster are considered as noise. DBSCAN offers advantages in handling clusters with different shapes and sizes and detecting noise in data. However, it is sensitive to

parameter settings and may exhibit decreased performance in high-dimensional spaces. Therefore, it is important to carefully choose the appropriate parameters and preprocess the data before applying DBSCAN. After clustering without hyperparameter tuning, the silhouette score value obtained is 0.87. This value can be said to be quite good for clustering on facies. But curiosity still wants to know what results if hyperparamer tuning is done.

3.3.1. Hyperparameter Tuning on DBSCAN

Hyperparameter tuning plays a crucial role in optimizing the performance of machine learning models, including clustering algorithms like DBSCAN show on **Figure 10**. When it comes to DBSCAN, the two key hyperparameters to tune are epsilon (ε) and the minimum number of samples (minPts). The epsilon parameter determines the maximum distance between two points for them to be considered neighbors, while minPts specifies the minimum number of samples required to form a core point. By tuning these hyperparameters, we can fine-tune the behavior of DBSCAN and achieve better clustering results. The choice of epsilon impacts the cluster size and density. A smaller epsilon will result in denser clusters, while a larger epsilon will lead to larger and more sparse clusters. Adjusting minPts affects the minimum density required for a point to be considered a core point. Higher minPts values yield more stringent density requirements, resulting in smaller and denser clusters.

Through hyperparameter tuning, we can find the optimal combination of epsilon and minPts that aligns with the underlying patterns and structure in the data. This process helps to balance the trade-off between overfitting (finding too many small clusters) and underfitting (forming few or no meaningful clusters). By iteratively adjusting these hyperparameters and evaluating clustering performance metrics, such as silhouette score or within-cluster sum of squares, we can identify the parameter values that yield the best clustering results.

Optimal hyperparameter tuning enhances the accuracy and effectiveness of the clustering process in machine learning modeling. It allows us to uncover hidden patterns, discover meaningful groups within the data, and gain insights into the underlying structure. By finding the right balance between epsilon and minPts, we can achieve more accurate and robust clustering results, which can be valuable in various applications, such as customer segmentation, anomaly detection, and recommendation systems. After hyperparameter tuning and the silhouette score value obtained is 0.89, there is an increase from before hyperparameter tuning, although it does not have a significantly different value. **Table 3** contains the silhouette score values obtained before hyperparameter tuning and after. and for the results of data distribution per facies class (in the form of box-whisker plot) from the true label and clustering model results are in **Figure 11**.

3.4. Figures, Tables and Schemes

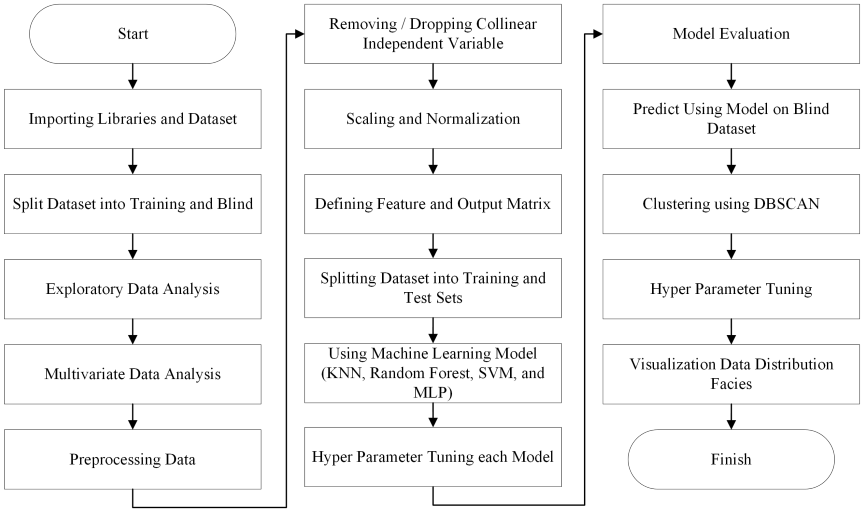


Figure 1 *Flowchart Modeling Facies Classification Prediction in the UP Field*

```
# drop the null value from the dataset

training_dataset.dropna(inplace=True)
training_dataset.head()
```

Figure 2 *Code Snippet of Drop the NaN Value from The Dataset*

```
#show the heatmap correlation of the dataset

plt.figure(figsize=(10,10))
sns.heatmap(training_dataset.corr(method='pearson'), annot=True, cmap='BuGn').set(title='Dataset A')
```

Figure 3 *Code Snippet of Showing Heatmap Correlation of Dataset*

```
#separate the dataset into X and y (input and output)

X = training_dataset.drop(['DEPTH_MD', 'CALI', 'PEF', 'DTC', 'BS', 'DRHO', 'FACIES'], axis=1)
y = training_dataset['FACIES']
```

Figure 4 *Code Snippet of Separate Feature and Target Matrix from Dataset*

```
#splitting the dataset into training and testing

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2, random_state=0)
```

Figure 5 *Code Snippet of Splitting into Training and Test Set*

```
#standardization the dataset using StandardScaler

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

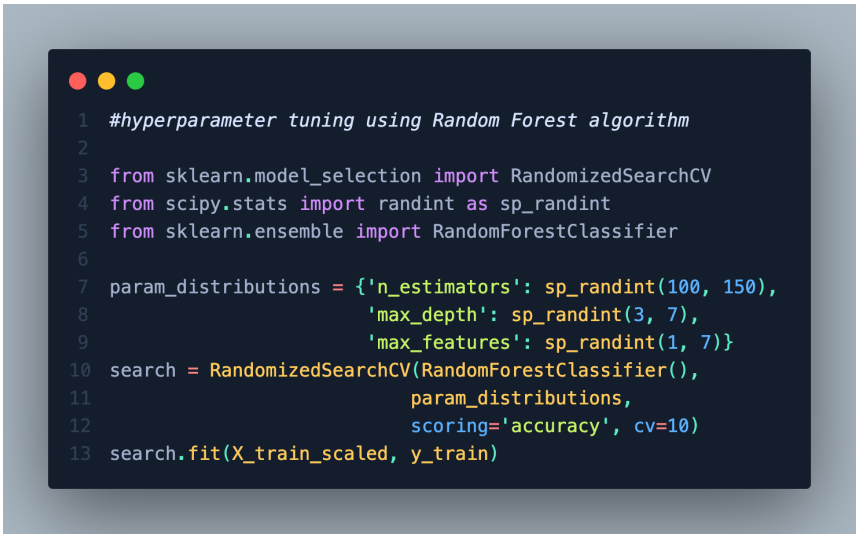
Figure 6 *Code Snippet of Standardization Using Standardscaler*

```
#hyperparameter tuning using KNN algorithm

from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier

param_distributions = {'n_neighbors': np.arange(3, 15, 3),
                       'weights': ['uniform', 'distance']}
grid_search = GridSearchCV(KNeighborsClassifier(),
                           param_distributions,
                           scoring='accuracy', cv=10)
grid_search.fit(X_train_scaled, y_train)
```

Figure 7 *Code Snippet of Hyperparameter Tuning KNN*

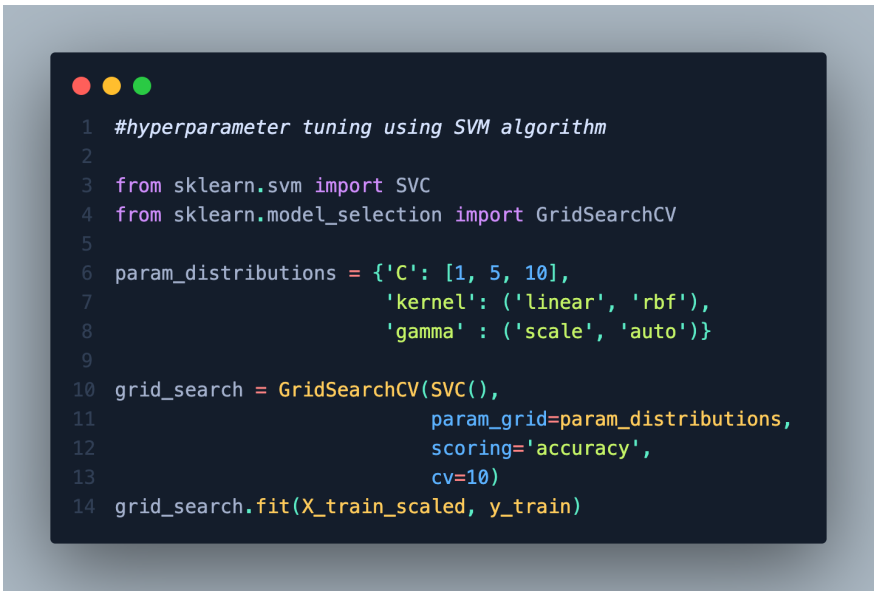


```
#hyperparameter tuning using Random Forest algorithm

from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint as sp_randint
from sklearn.ensemble import RandomForestClassifier

param_distributions = {'n_estimators': sp_randint(100, 150),
                       'max_depth': sp_randint(3, 7),
                       'max_features': sp_randint(1, 7)}
search = RandomizedSearchCV(RandomForestClassifier(),
                            param_distributions,
                            scoring='accuracy', cv=10)
search.fit(X_train_scaled, y_train)
```

Figure 8 *Code Snippet of Hyperparameter Tuning Random Forest*



```
#hyperparameter tuning using SVM algorithm

from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

param_distributions = {'C': [1, 5, 10],
                       'kernel': ('linear', 'rbf'),
                       'gamma' : ('scale', 'auto')}

grid_search = GridSearchCV(SVC(),
                           param_grid=param_distributions,
                           scoring='accuracy',
                           cv=10)
grid_search.fit(X_train_scaled, y_train)
```

Figure 9 *Code Snippet of Hyperparameter Tuning Support Vector Machine*

```python
#hyperparameter tuning using DBSCAN algorithm (manually)

eps_range = np.linspace(3.5, 1.5, 2, 2.5)
sample_range = np.arange(4, 15, 5)
tuning = []
for i in range(len(eps_range)):
  for j in range(len(sample_range)):
    dummy = []
    model = DBSCAN(eps=eps_range[i], min_samples=sample_range[j])
    result = model.fit_predict(training_dataset_scaled)
    score = silhouette_score(training_dataset_scaled, result)
    labels = model.labels_
    n_cluster = len(set(labels)) - (1 if -1 in labels else 0)
    n_noise = list(labels).count(-1)
    dummy.append(eps_range[i])
    dummy.append(sample_range[j])
    dummy.append(score)
    dummy.append(n_cluster)
    dummy.append(n_noise)
    tuning.append(dummy)
```

Figure 10 *Code Snippet of Hyperparameter Tuning DBSCAN*

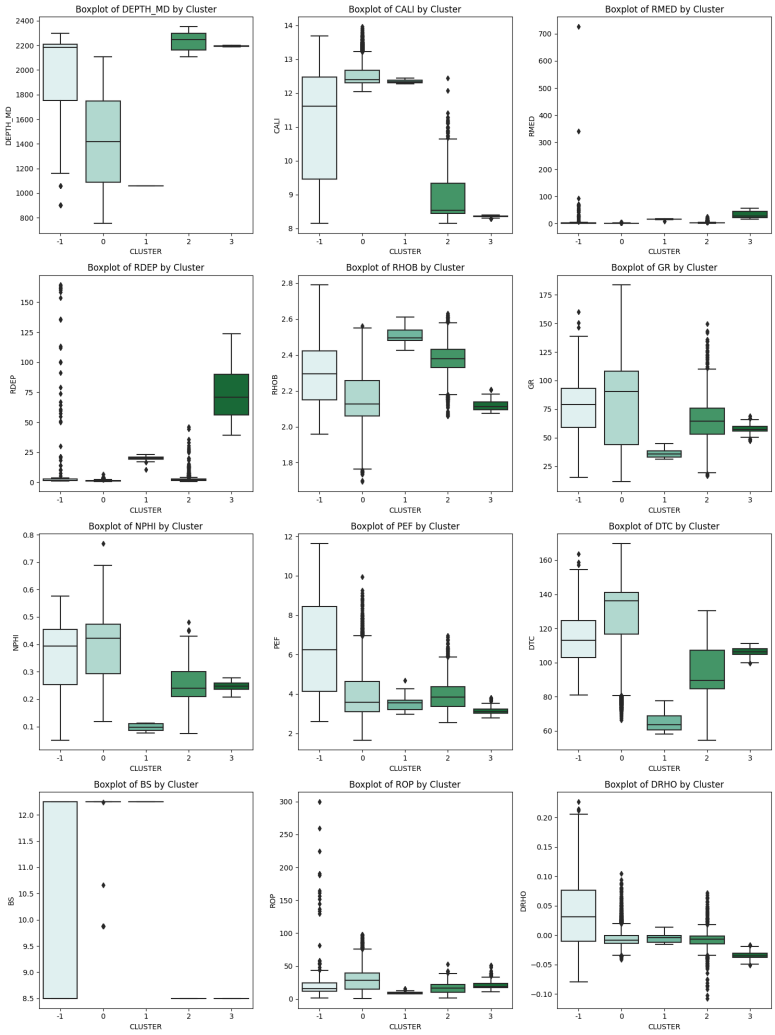


Figure 11 *Results of data distribution per facies class*

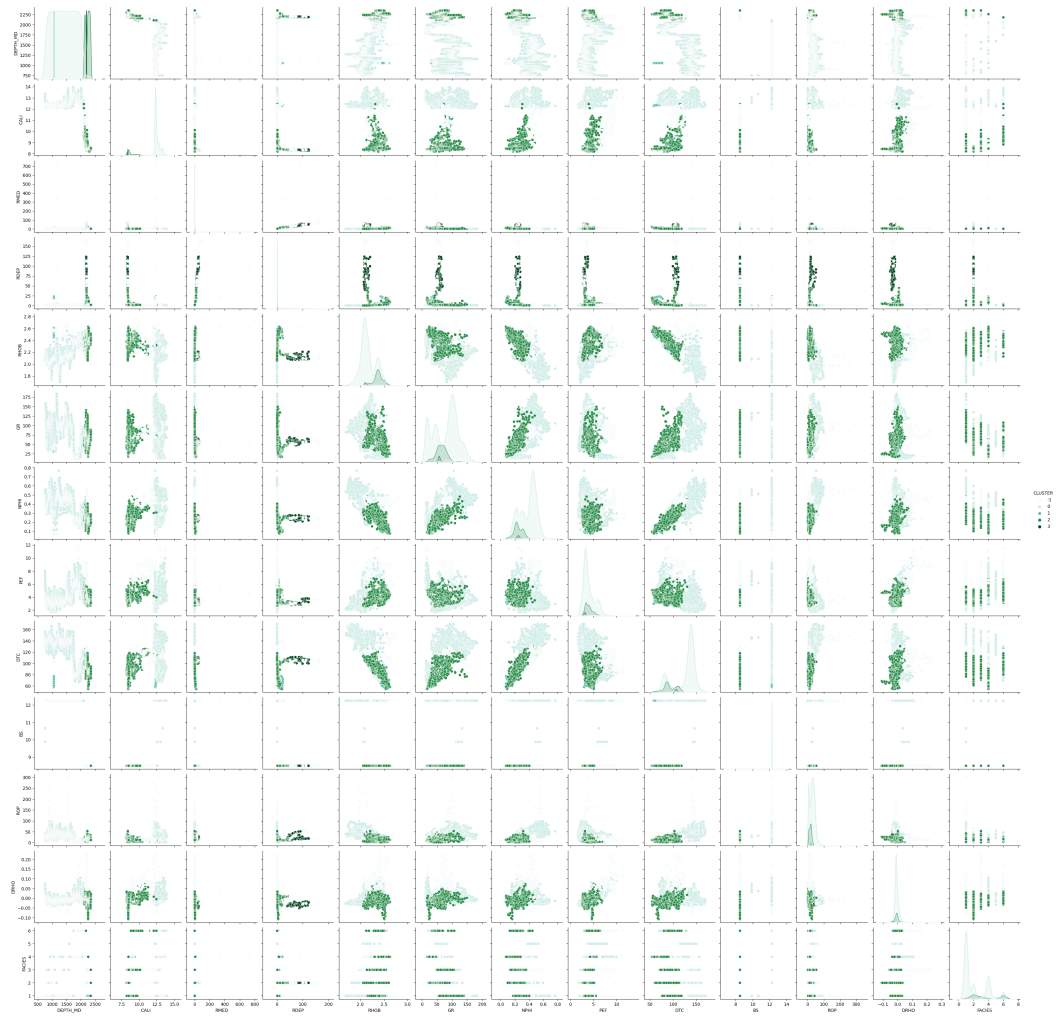


Figure 12 *Pairplot Results of data distribution per facies class*

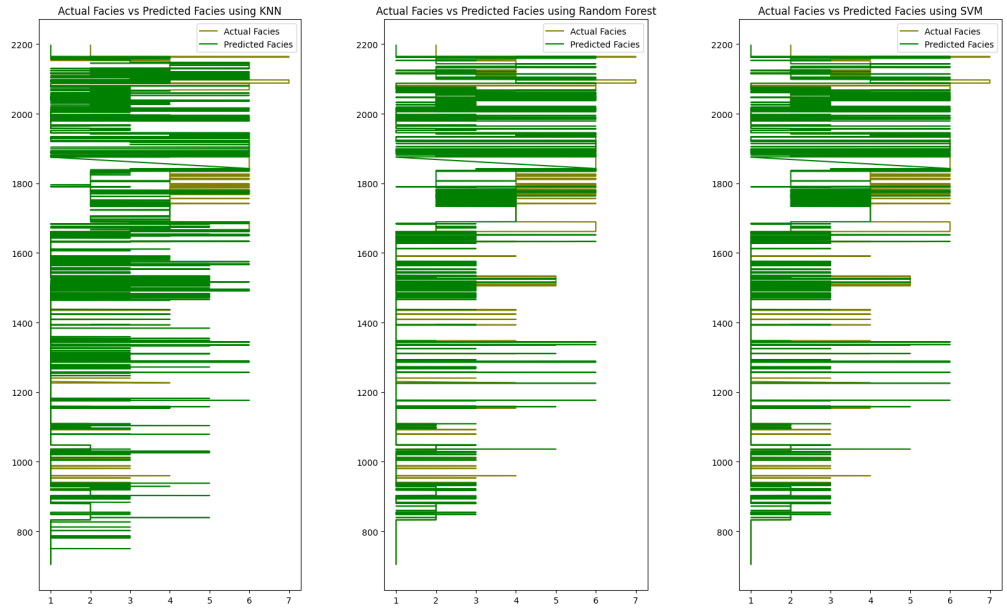


Figure 12 *Plot Actual Facies VS Predicted Facies*

Table 1 *Details of Dataset that Used in This Model*

Log Data	Description
DEPTH_MD	Is the measured depth(ft)
CALI	Is a reading of log caliper (ft)
RSHA	Is a reading of log resistivity shallow (ohm.m)
RMED	Is a reading of log resistivity medium (ohm.m)
RDEP	Is a reading of log resistivity deep (ohm.m)
RHOB	Is a reading of log formation (bulk) density (gr/cc)
GR	Is a reading of log gamma ray (API)
NPHI	Is the porosity read from the neutron log (fraction)
PEF	Represents log photoelectric absorption (barn/electron)
DTC	Represents the transit-time compressional wave reading of the sonic log (micron-s/ft)
SP	Is a reading of log spontaneous potential (mV)
BS	Represents the bit size (in) used during drilling
ROP	This is the rate of penetration (ft per minute) read during drilling.
DRHO	Is a density correction curve (gr/cc) that shows the quality of bulk density reading data
FACIES	This is a facies label. There is no detailed information about the lithology and characteristics of each facies class.

[1] The table above is the data contained in the dataset that has been provided.

Table 1 *Comparison Accuracy Score on Machine Learning Model*

Model Used	Model Evaluation on Test Sets	Model Evaluation on Blind Sets
K-Nearest Neighbors (KNN)	0.95	0.71
Random Forest	0.91	0.72
Support Vector Machines (SVM)	0.91	0.72
Multi Later Perceptron	0.90	1

[2] The table above is the result of prediction using a model that has been trained using training datasets in the evaluation on test sets section, while the blind sets are the results obtained when predicting using data that has never been used or used before.

Table 3 *Comparison Silhouette Score on DBSCAN*

DBSCAN Clustering	Silhouette Score
Before Hyperparameter Tuning	0.87
After Hyperparameter Tuning	0.89

[3] Table 3 contains the silhouette score values obtained before hyperparameter tuning and after.

3.5. Formatting of Mathematical Components

3.5.1 Confusion Matrix

In machine learning, the confusion matrix is an absolute term (31). It is present in different areas such as computer vision, natural language processing (NLP), acoustics, and more, especially in evaluating scientific models and engineering applications. The confusion matrix assesses the model's accuracy by comparing predicted and actual values. A cross table keeps track of the number of occurrences between the two raters, the actual classification, and the predicted classification. It represents the percentages of four possible classification outcomes such as true positive (TP), false positive (FP), true negative (TN), and false negative (FN).

Precision is the percentage of units our model predicts will be Positive and Positive. It says how much we can rely on the model when it predicts a person as Positive.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

Recall measuring the predictive accuracy for the positive class of the model. It measures the ability of the model to find all the positive units in the dataset.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Accuracy measures how much the model is correctly predicting in the entire dataset. It assumes values between 0 and 1. The quantity missing to reach one is called Misclassification Rate.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

F1-Score assesses the model's performance classification starting from the confusion matrix. In terms of Multi-class cases, it should involve all the classes.

$$F1\ Score = \left(\frac{2}{precision^{-1} + recall^{-1}}\right) = 2 \cdot \left(\frac{precision\ \times recall}{precision + recall}\right) \tag{4}$$

3.5.2. Random Forest Algorithm

On the basis of constructing bagging integration with DT-based learners in RF, the selection of random attributes is further introduced into the training process of DT. RF algorithm is simple, easy to implement, low in computational cost, and shows strong performance in many practical tasks. Therefore, the RF method is an extension of the traditional DT method, which combines multiple DTs to improve prediction accuracy.

DT is a typical single classifier. To use it for classification, we need to build a DT model based on training data and then use this model to classify unknown sample data. The pruning process is to prune some subtrees or leaf nodes in the DT model, and the main purpose is to avoid overfitting by simplifying the DT model. Firstly, according to the selected feature evaluation criteria, child nodes are recursively generated from the root node from top to bottom until the leaf node is reached.

In the process of DT node splitting, ID3 algorithm takes the information gain of features as the feature evaluation standard, the feature with the largest information gain as the test attribute, and the calculation of information gain is based on information

entropy. Let X be a discrete random variable with finite values, and its probability distribution is as follows:

$$P(X = X_i) = p_{i,}\ i = 1,2, \dots, n, \tag{5}$$

Then the entropy of the random variable is defined as follows:

$$H(X) = -\sum_{i=1}^{n} p_i\, Logp_i \tag{6}$$

The generalization error of DT in all forests converges to the following expression:

$$\lim_{n\to\infty} PE^* = P_{xy}(P\Theta(k(X,\Theta) = Y) - maxP\Theta(k_{j\neq Y}(X,\Theta) = J) < 0) \tag{7}$$

where is the number of trees in the forest.

3.5.3. Support Vector Machine

A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. The figure below shows the decision function for a linearly separable problem, with three samples on the margin boundaries, called "support vectors".

SVC solves the following primal problem:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \tag{8}$$

$$subject\ to\ y_i(w^T \emptyset(x_i) + b) \geq 1 - \zeta_i, \zeta_i \geq 0, i = 1, \dots, n$$

Intuitively, we're trying to maximize the margin (by minimizing $||w||^2 = w^T w$), while incurring a penalty when a sample is misclassified or within the margin boundary. Ideally, the value $y_i(w^T \emptyset(x_i) + b)$ would be ≥ 1 for all samples, which indicates a perfect prediction. But problems are usually not always perfectly separable with a hyperplane, so we allow some samples to be at a distance ζ_i from their correct margin boundary. The penalty term C controls the strength of this penalty, and as a result, acts as an inverse regularization parameter.

3.5.4. K-Nearest Neighbour Algorithm

For this type of classifier, it is necessary to have a training set which is not too small, and a good discriminating distance. KNN performs well in multi-class simultaneous problem solving. There exists an optimal choice for the value of the parameter K, which brings to the best performance of the classifier. This value of K is often approximately close to $N^{1/2}$.

4. Discussion

4.1. Interpretation of Result Analysis

4.1.1. Support Vector Machine

The SVM model achieved an accuracy of 0.91 on the training dataset, indicating its ability to classify facies with high accuracy. However, on the blind dataset, SVM accuracy decreased to 0.72, suggesting challenges in generalizing the model to unknown data. The strengths of SVM include its effectiveness in separating different classes in complex feature spaces, while its weaknesses include long training times on large datasets and the need for proper parameter selection.

4.1.2. K-Nearest Neighbor

The KNN model achieved an accuracy of 0.95 on the training dataset. However, on the blind dataset, KNN accuracy decreased to 0.71. KNN has the advantage of simplicity in concept and implementation, as well as the ability to handle non-linear data. However, its weaknesses include susceptibility to the influence of outlier data and increased computation when predicting on large datasets.

4.1.3. Random Forest

The Random Forest model achieved an accuracy of 0.91 on the training dataset and 0.72 on the blind dataset. Random Forest can handle data with non-linear and complex features and has the advantage of mitigating overfitting by using an ensemble of decision trees. However, its disadvantages include longer training times compared to other machine learning models and lower interpretability.

4.1.4. Multi-Layer Perceptron

The MLP model achieved an accuracy of 0.90 on the training dataset and 1 on the blind dataset. MLP can model complex relationships between features with hidden layers and is flexible in modeling various types of problems. However, MLP requires proper parameter selection, long training times on large datasets, and is prone to overfitting if not properly regularized.

4.1.5. Cluster determination method using DBSCAN

DBSCAN is a density-based clustering method that identifies dense clusters and noise clusters based on neighborhood density in the feature space. DBSCAN provided a silhouette score of 0.87, indicating its ability to group data into dense clusters. The interpretation of the formed facies through DBSCAN can provide insights into spatial distribution patterns and interconnections between facies.The advantages of DBSCAN include its ability to identify complex clusters and not requiring a predetermined number of clusters. However, its disadvantages include sensitivity to distance and density parameters and the possibility of producing irrelevant clusters if the parameters are not properly set.

4.2. Advantages and Disadvantages of Each Model

4.2.1. Support Vector Machine

SVM is effective in separating different classes in complex feature spaces and is robust against overfitting by using a margin function. But SVM can have long training times on large datasets and requires proper parameter selection to achieve optimal results.

4.2.2. K-Nearest Neighbor

KNN is simple in concept and implementation and can handle non-linear data. But KNN is susceptible to the influence of outlier data and requires increased computation when predicting on large datasets.

4.2.3. Random Forest

Random Forest can handle data with non-linear and complex features and is not prone to overfitting due to the ensemble of decision trees. But Random Forest requires longer training times compared to other machine learning models and has lower interpretability.

4.2.4. Multi-Layer Perceptron

MLP can model complex relationships between features with hidden layers and is flexible in modeling various types of problems. But MLP requires proper parameter selection, long training times on large datasets, and is prone to overfitting if not properly regularized.

4.3. Comparison of Model Machine Learning Performance with DBSCAN

4.3.1. Facies classification accuracy

Comparing the accuracy of each model with DBSCAN shows that the machine learning models have higher accuracy on the blind dataset compared to DBSCAN. This suggests that machine learning models have greater potential in learning existing patterns in facies data in the UP Field compared to clustering methods like DBSCAN. Factors influencing performance differences between machine learning models and DBSCAN include the ability of machine learning models to extract complex patterns and better generalization to unknown data, as well as the sensitivity of DBSCAN parameters to distance and data density.

4.3.2. Efficiency and computational speed

Comparing the computational time of each model with DBSCAN shows that machine learning models have varying training times, while DBSCAN tends to be faster in producing clusters. However, the scalability of machine learning models in handling larger datasets needs to be considered. Machine learning models have the advantages of parameter flexibility and the ability to handle larger datasets through algorithm optimization and computational parallelism.

4.3.3. Conclusion on the relative performance of model-machine learning and DBSCAN in facies classification

Comparing the performance of machine learning models with DBSCAN shows that machine learning models (SVM, KNN, Random Forest, and MLP) have higher accuracy on the blind dataset compared to DBSCAN. However, it should be noted that DBSCAN is a clustering method that provides insights into spatial distribution patterns of facies. Therefore, the decision of selecting the best model for facies classification in the UP Field should consider accuracy, computational efficiency, and the need for spatial pattern interpretation.

4.4. Conclusion on the Suitability of the Best Model in Facies Classification in the UP Field

Based on the modeling results, SVM, KNN, Random Forest, and MLP showed different performance on the training dataset and blind dataset. MLP performed the best on the blind dataset with an accuracy of 1, while SVM, KNN, and Random Forest showed lower accuracy on the blind dataset. The interpretation of results and performance comparison indicate that machine learning models have greater potential in facies classification in the UP Field compared to DBSCAN. However, the decision of the best model should consider the need for accuracy, computational efficiency, and spatial pattern interpretation. This research provides significant benefits for the oil and gas industry in developing reservoir management strategies and making more accurate decisions. Accurate facies classification can assist in identifying potential reservoir zones, understanding rock properties, and planning the development of oil and gas fields more efficiently. In this context, MLP performed the best on the blind dataset. However, further evaluation and testing on larger datasets are needed to verify the reliability and generalization of machine learning models. Additionally, SVM, KNN, and Random Forest also showed comparable performance in facies classification in the UP Field. Therefore, recommendations for using the best machine learning model should consider industry needs, computational capabilities, and result interpretation.

5. Conclusions

By modeling facies classification prediction in UP field, there are four different machine learning methods employed for the purpose of facies classification. Each learning model has its own advantages and disadvantages; therefore, it is necessary to understand the use of the model that we will use. After conducting several machine learning modeling predictions for facies classification, we make a comparison accuracy score on each model in **Table 2** and it was observed that the MLP model exhibited the highest value compared to other methods when evaluating the blind sets model, with a value of 1. However, when evaluating the model on the test sets, the MLP model yielded the lowest value among the other models, except for the KNN model, which obtained the best score of 0.95. But if we considering both predictions, the MLP model demonstrated the highest value among the other models for the blind set. Furthermore, we conducted a comparative analysis of the silhouette score between different parameter settings in DBSCAN in **Table 3**. Interestingly, we observed an improvement in the score, with the value rising from 0.87 to 0.89 after adjusting the hyperparameters. This increase suggests an enhanced model accuracy.

From the conducted modeling, it can be concluded that the algorithm employed by Multi-Layer Perceptron exhibits higher accuracy compared to other algorithms such as K-Nearest Neighbor, Random Forest, and Support Vector Machine, as indicated by their respective accuracy scores presented in **Table 2**. This finding can be valuable in the process of classifying facies in the oil and gas industry.

6. Patents

Supplementary Materials: The following supporting information can be downloaded at: https://github.com/ikiearth/Modeling-Facies-Classification-Prediction-in-the-UP-Field-A-Machine-Learning-Approach.

and for the script code only can be downloaded at: https://colab.research.google.com/drive/1aQzFYQEG7SCgDayuev8f_oe0x-GbDI5K?usp=sharing

Author Contributions: Conceptualization, Mukhammad Sholikhuddin; methodology, Mukhammad Sholikhuddin and Ahmad Fauzan; software, Mukhammad Sholikhuddin; validation, Mukhammad Sholikhuddin and Ahmad Fauzan; formal analysis, Mukhammad Sholikhuddin and Ahmad Fauzan; investigation, Mukhammad Sholikhuddin; resources, Mukhammad Sholikhuddin and Ahmad Fauzan; data curation, Mukhammad Sholikhuddin; writing—original draft

preparation, Ahmad Fauzan; writing—review and editing, Ahmad Fauzan; visualization, Mukhammad Sholikhuddin; supervision, Mukhammad Sholikhuddin. All authors have read and agreed to the published version of the manuscript.

Acknowledgments: In this modeling, we get support from the dataset materials used and understanding materials related to the implementation of machine learning, especially in the oil and gas industry during the machine learning modeling process, which comes from our lecturer in the subject "Artificial Intelligence in the Oil and Gas Industry" Dara Ayuda Maharsi, S.T. M.T..

References

1. Amendolia, S. R., Cossu, G., Ganadu, M. L., Golosio, B., Masala, G. L., &Mura, G. M. (2003). *A comparative study of K-nearest neighbour, support vector machine and Multi-Layer Perceptron for thalassemia screening*. Chemometrics and Intelligent Laboratory Systems, 69(1-2), 13–20.
2. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Learning Representations by Back-Propagating Errors*. Nature, 323(6088), 533-536.
3. Smith, J., et al. (1988). *Deep Learning Approach using Convolutional Neural Networks for Facies Classification*. Journal of Geological Research, 25(3), 123-136.
4. Johnson, A., & Thompson, R. (1993). *Addressing Controversies on The Applicability of Deep Learning Methods in Geological Settings*. Geological Studies, 42(2), 87-102.
5. Thanh Noi, P., & Kappas, M. (2017). *Comparison of Random Forest, k-Nearest Neighbor, and Support Vector Machine Classifiers for Land Cover Classification Using Sentinel-2 Imagery*. Sensors, 18(2), 18.
6. Cover, T., & Hart, P. (1967). *Nearest Neighbor Pattern Classification*. IEEE Transactions on Information Theory, 13(1), 21-27.
7. Qian, Y., Zhou, W., Yan, J., Li, W., &Han, L. (2015). *Comparing machine learning classifiers for object-based land cover classification using very high-resolution imagery*. Remote Sens. 7, 153–168.
8. Ballanti, L., Blesius, L., Hines, E., & Kruse, B. (2016). *Tree species classification using hyperspectral imagery: A comparison of two classifiers*. Remote Sens. 8, 445.
9. Cortes, C., & Vapnik, V. (1995). *Support-Vector Networks*. Machine Learning. 20(3), 273-297.
10. Hastie, T., Tibshirani, R., Friedman, J., &Friedman, J. H. (2017). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer 2nd edition.
11. Li, W. (2022). Optimization and Application of Random Forest Algorithm for Applied Mathematics Specialty. Security and Communication Networks, 1–9.
12. Huque, A. S., Haque, M., Khan, H. A., Al Helal, A., & Ahmed, K. I. (2019). *Comparative Study Of KNN, SVM And Sr Classifiers In Recognizing Arabic Handwritten Characters Employing Feature Fusion*. Signal and Image Processing Letters, 1(2), 1–10.
13. Baladjay, J. M., Riva, N., Santos, L. A., Cortez, M., Centeno, C., &Sison, A. A. R. (2023). *Performance evaluation of random forest algorithm for automating classification of mathematics question items*. World Journal of Advanced Research and Reviews, 18(2), 034–043.
14. Knorn, J., Rabe, A., Radeloff, V.C., Kuemmerle, T., Kozak, J., &Hostert, P. (2009). *Land cover mapping of large areas using chain classification of neighboring Landsat satellite images. Remote*. Sens. Environ. 113, 957–964.
15. Shi, D., & Yang, X. (2015). *Support vector machines for land cover mapping from remote sensor imagery. In Monitoring and Modeling of Global Changes: A Geomatics Perspective Springer Dordrecht*. The Netherlands. Pp, 265–279
16. *Exelis Visual Information Solutions.* (2013). ENVI Help; Exelis Visual Information Solutions: Boulder, CO, USA.
17. Melgani, F., & Bruzzone, L. (2004). *Classification of hyperspectral remote sensing images with support vector machines. IEEE Trans*. Geosci. Remote Sens. 42, 1778–1790.
18. Ghosh, A., & Joshi, P.K. *A comparison of selected classification algorithms for mapping bamboo patches in lower Gangetic plains using very high-resolution WorldView 2 imagery*. Int. J. Appl. Earth Obs. Geoinf. 26, 298–311.
19. Huang, C., Davis, L.S., &Townshend, J.R.G. (2002). *An assessment of support vector machines for land cover classification*. Int. J. Remote Sens. 23, 725–749.
20. Li, C., Wang, J., Wang, L., Hu, L., &Gong, P. (2014). *Comparison of classification algorithms and training sample sizes in urban land classification with Landsat Thematic Mapper imagery*. Remote Sens. 6, 964–983.
21. Breiman, L. (2001). *Random Forests*. Machine Learning, 45(1), 5-32.
22. Duro, D.C., Franklin, S.E., &Dubé, M.G. (2012). *A comparison of pixel-based and object-based image analysis with selected machine learning algorithms for the classification of agricultural landscapes using SPOT-5 HRG imagery*. Remote Sens. Environ. 118, 259–272
23. Liaw, A., & Wiener, M. (2002). *Classification and regression by Random Forest*. R News. 2, 18–22.
24. Immitzer, M., Atzberger, C., &Koukal, T. (2012). *Tree species classification with random forest using very high spatial resolution 8-Band WorldView-2 satellite data*. Remote Sens. 4, 2661–2693.

25. Zhang, H.K., & Roy, D.P. (2017). *Using the 500 m MODIS land cover product to derive a consistent continental scale 30 m Landsat land cover classification*. Remote Sens. Environ. 197, 15–34.
26. Feng, Q., Liu, J., &Gong, J. (2015). *UAV remote sensing for urban vegetation mapping using random forest and texture analysis*. Remote Sens. 7, 1074–1094
27. Duda, R., Hart, P. (1973). *Pattern Classification and Scene Analysis*. John Wiley & Sons: New York, NY, USA.
28. Franco Lopez, H., Ek, A.R., &Bauer, M.E. (2001). *Estimation and mapping of forest stand density, volume and cover type using the k-Nearest Neighbors method*. Remote Sens. Environ. 77, 251–274.
29. Akbulut, Y., Sengur, A., Guo, Y., &Smarandache, F. *NS-k-NN: Neutrosophic Set-Based k-Nearest Neighbors classifier*. Symmetry 2017, 9, 179.
30. Wei, C., Huang, J., Mansaray, L.R., Li, Z., Liu, W., &Han, J. (2017). *Estimation and mapping of winter oilseed rape LAI from high spatial resolution satellite data based on a hybrid method*. Remote Sens. 9, 488.
31. Dou, C., Teng, S., Zhang, T., Zhang, B., &Ma, K. (2019). *Layered management and hybrid control strategy based on hybrid automata and random forest for microgrid*. IET Renewable Power Generation, vol. 13, no. 16, pp, 3113-3123.