Implementation of Random Forest for Lithofacies Classification

I. Introduction

Lithofacies identification is a process that allows the determination of the hydrocarbon bearing zone. So, the potential resource can be developed and produced. The ideal sources for lithofacies classification are core samples of rock extracted from wells within the field. However, core samples are not always available due to the associated cost. Indirect measurement such as well logging emerges as an alternative method to classify facies. Nevertheless, the interpretation of these well logs is a mammoth task which highly needs experience expert to interpret the measurement and convert it to the meaningful data. The application of machine learning is a promising method facilitate the lithofacies classification process.

The objective of this capstone project is to investigate the performances of variety machine learning model, so the best model can be determined.

II. Data Acquisition and Cleaning

The data used in this project is obtained from Alberta Geological Survey. it is a data collection of 2193 wells to map the McMurray Formation and the overlying Wabiskaw Member of the Clearwater Formation in the Athabasca Oil Sand Area. The obtained data was organized, and cleaned, so minimal data cleaning is performed for this project.

- Two csv files “Picks” and “Intellog” were loaded into Pandas Dataframes. The columns of “Picks” dataframe were renamed to “SitID”, “HorID”, “Depth” and “Quality”.
- The “Depth” column was converted from object to numeric data type
- The “RW” column of “Intellog” dataframe was converted to numeric data type
- Merged “Intellog” and “Picks” dataframe together to become “Main_File” dataframe by inner joint method by the common columns “SitID” and “Depth”
- Dropped any N/A values in “Main_File”
- Created features matrix by dropping columns “SitID”, “HorID”, “Depth” and “LithID” from “Main_File” dataframe
- Finally, created target column by choosing column “LitID” only from “Main_File” Dataframe

III. Data Exploration

After “Picks” and “Intellog” CSV were loaded into dataframes, a series of explorational steps were performed to understand the dataset. Df.info() were used to quickly determined the size and the type of data as shown in **Fig.1** below

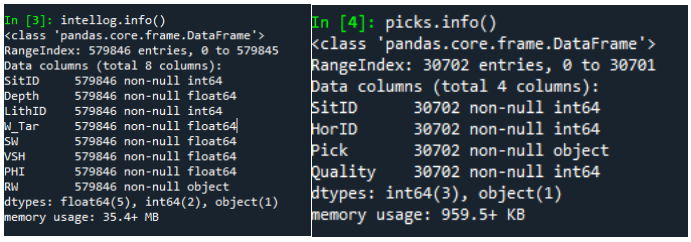


Figure 1: The quick glance into Intellog and Picks Dataframes

As shown, there were 579,846 values in each column of "Intellog" dataframe. No NaN values were present in this dataframe. Columns "SitID", "LithID" has data type as int64. Columns "Depth", "W_Tar", "SW","VSH", and "PHI" were in float64 format. However, column "RW" are in non-null object. So, this column "RW" should be converted to numeric data type instead. Similarly, dataframe "Picks" showed 30,702 values in each column. No NaN values were present neither. Columns "SitID", "HorID" and "Quality" were in int64 format while "Pick" columns were in non-null object. So, it must be converted to numeric data type. In addition, the "Pick" column indicates the depth where the rock samples were retrieved; hence, it can be renamed to "Depth" columns.

Next, the two dataframes were merged by columns "SitID" and "Depth". As shown in **Fig.2**, there were 10,318 values in each column, except column "RW" which only had 10,204 values. Therefore, the NaN can be dropped from dataframe.

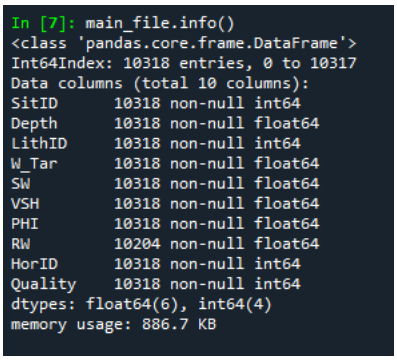


Figure 2: The overview of the main dataframe used in this project

The data distribution can also be viewed by using box plots. As illustrated in **Fig.3**, majority of features have outliers in its data except for "PHI" column. Further investigation on these outliers should be carried out to understand why these outliers existed. For this project, these outliers were left alone .

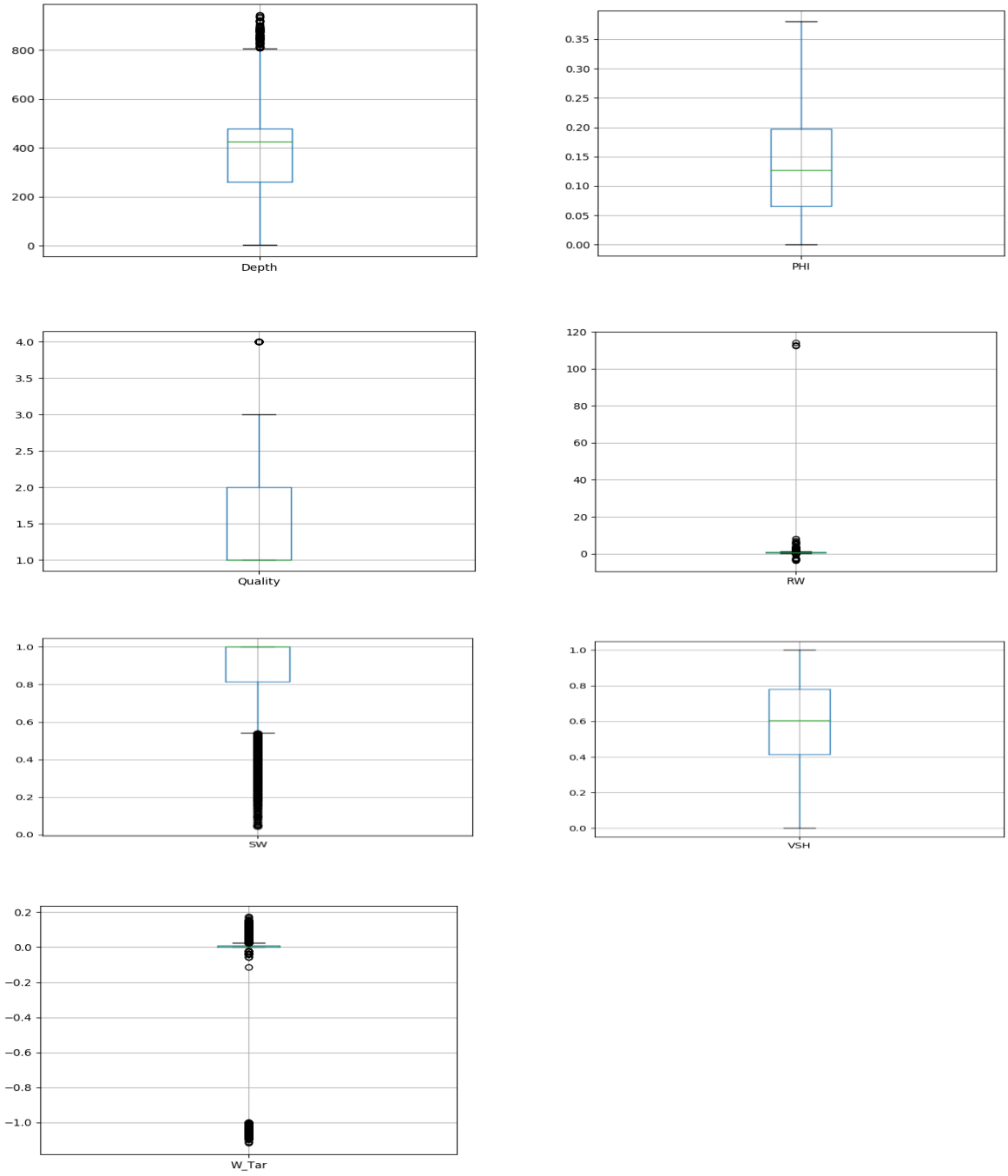


Figure 3: The boxplots to show the distribution of each feature in this dataset

Lithofacies classification is the target of this project. It is necessary to see how different facies distributed within this dataset. As shown from the **Fig.4**, the proportions of classes 1,2,4, and 5 are 12.95%, 19.59%,28.40%, and 36.52% respectively. In contrast, classes 0,3, and 6 shows 0.20%, 1.96%, and 0.39%. Therefore, this dataset is highly imbalanced.

```
Class=2, Count=1999, Percentage= 19.59%
Class=4, Count=2898, Percentage= 28.40%
Class=1, Count=1321, Percentage= 12.95%
Class=5, Count=3726, Percentage= 36.52%
Class=3, Count=200, Percentage= 1.96%
Class=0, Count=20, Percentage= 0.20%
Class=6, Count=40, Percentage= 0.39%
```

Figure 4: The target distribution of this dataset demonstrated an highly imbalanced data

A correlation heat map is constructed to examine the correlation between features. Several correlations were observed from the **Fig.5** below

- Strong correlation between volume of shale and lithology
- Strong negative correlation between volume of shale and porosity
- Strong correlation between porosity and lithology
- Good correlation between water saturation and volume of shale
- Good negative correlation between porosity and water saturation

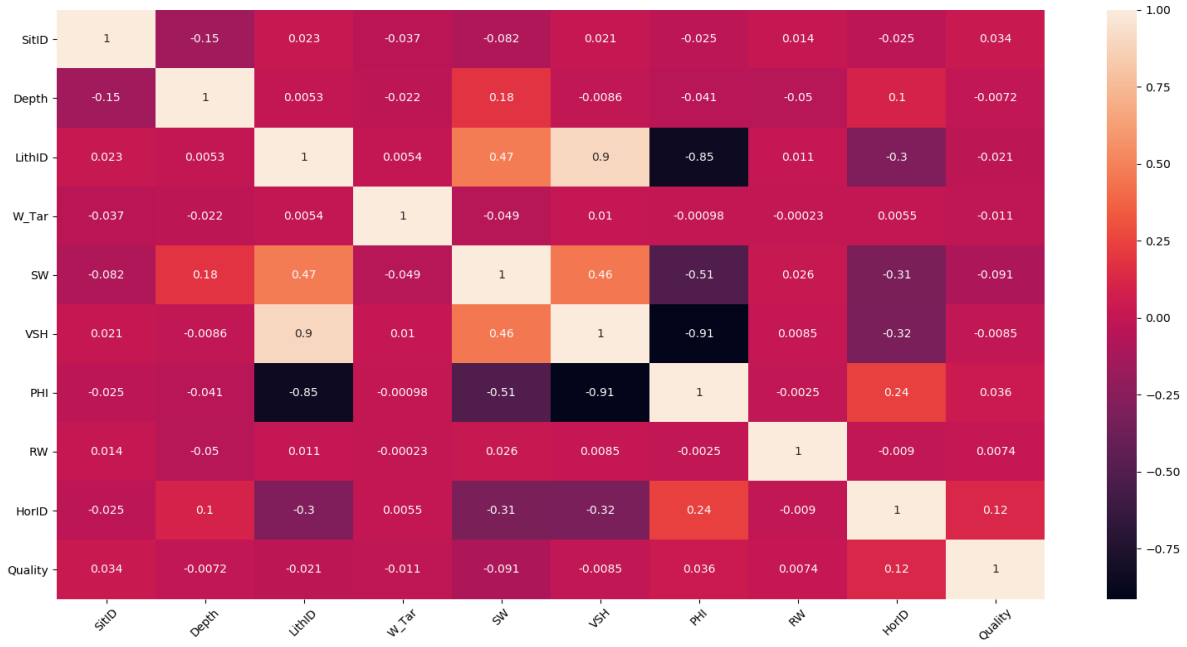
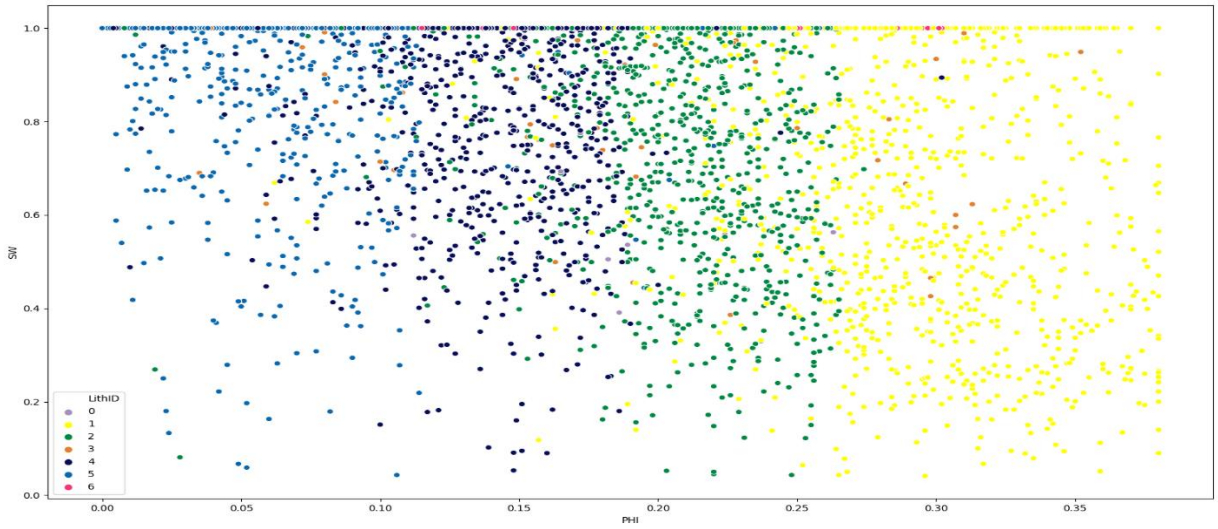


Figure 5: The correlation heatmap between features and target

From the observed correlation above, I am wondering how lithology classes distribution look like when these parameters are plotted against each other as illustrated in **Fig.6**



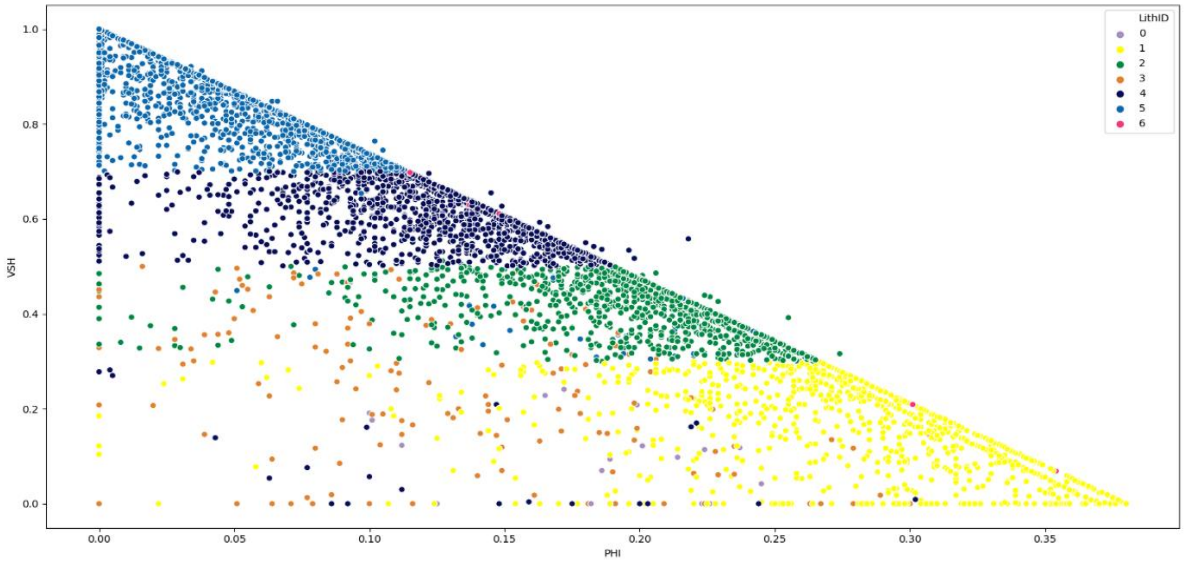


Figure 6: Lithofacies clusters are distinguishable for majority classes. Minority classes are blended into these majority classes clusters.

As **Fig.6** illustrated, lithology classes 1,2,4, and 5 are distinct clusters that the machine learning model can classify them with high degree of accuracy. Nevertheless, lithology class 0,3 and 6 are mixed into other classes as expected because the rock that belong to these classes have mixed properties of class 1,2,4 and 5. Furthermore, the proportion of samples for class 0,3 and 6 are significantly smaller than the proportion for other classes. Therefore, it will be challenging for the model to have good performance to predict the class 0,3 and 6 correctly.

IV. In-Depth Analysis and Results

1. Machine Learning Models Screening

After data exploratory, the features were standardized to have zero mean and unity variance. Next, dataset was split into training and holdout set in the of fashion of 70%:30%. K-fold cross validation with 10 folds and 3 times repetition was implemented to the data to validate the model's performance. Then, a variety of machine learning models with default parameters were trained and fitted the data. Macro F1 score was chosen as the metric for this imbalanced and multi-classification problem. Macro F1 score is well-suited metric because it put equal weight to each class, so it can capture how well the model performs to predict minority classes. Otherwise, if weighted-average F1 score is used, a nearly perfect F1 score will be observed which failed to validate the model for predicting minority classes. A boxplot as illustrated in **Fig.7,** showcases the performance of each model. It appears the ensemble learning models (included Random Forest, Gradient Boosting and Stacking) outperformed other models.

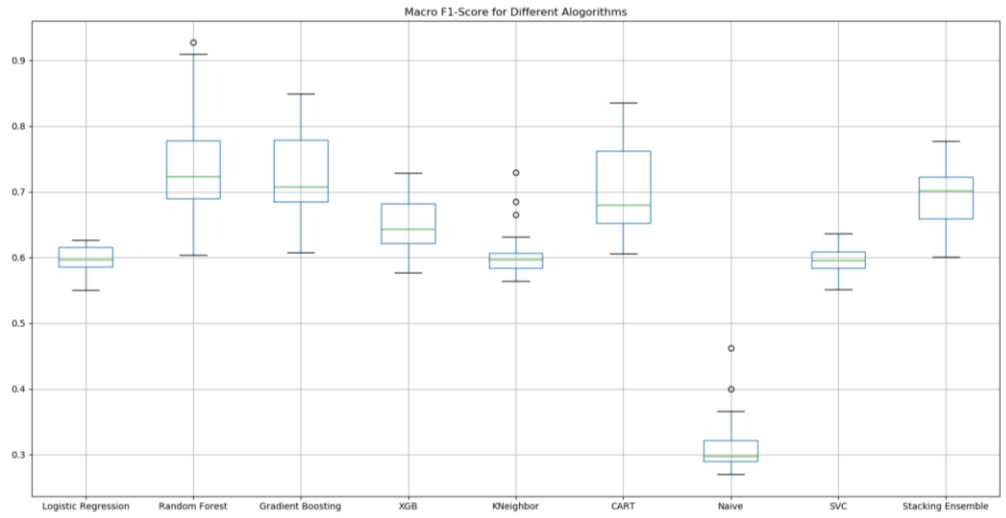


Figure 7: Machine Learning algorithm screening to show the promising model for this dataset

2. Random Forest Overview

Bagging method derived its name from "Bootstrap Aggregating". The idea of bagging is to fit several independent models and take an average of their predictions to obtain a better model with lower variance. It is nearly impossible in practice to achieve because it would require lots of data which often are not available. Therefore, bootstrap is an attractive solution because under some assumptions, these bootstrap sample can be considered as representative and independent samples of true data distribution. Bagging method can be divided into two steps:

- Multiple bootstrap samples with size N are created that acts as another independent and identically distributed
- Multiple models are trained and fit these bootstrap samples and finally aggregate their results by taking average.

Because bootstrap samples are independent and identically distributed, averaging multiple models do not change the expected result but it only reduces its variance.

Random forest is a bagging method in a sense that its uncorrelated trees fit on the bootstrap samples and their results are combined to produce a better model with lower variance. Nevertheless, random forest uses another trick to make its tree less correlated to each other; thus, enhance its performance by sampling over features and keep only a random subset of them to build the tree.

3. Workflow

An overview of workflow is demonstrated in the **Fig.9** below. The processed data was split into Dataset and Holdout set. Dataset then subjected to K-Fold cross validation to split into training and validation sets. Random Forest was used to train and fit training dataset. Because the dataset in this project is highly imbalanced and multiclass, different resampling techniques were implemented to see if they can enhance the performance of the models. Finally, random grid search and grid search were carried out to tune models' hyper-parameters.

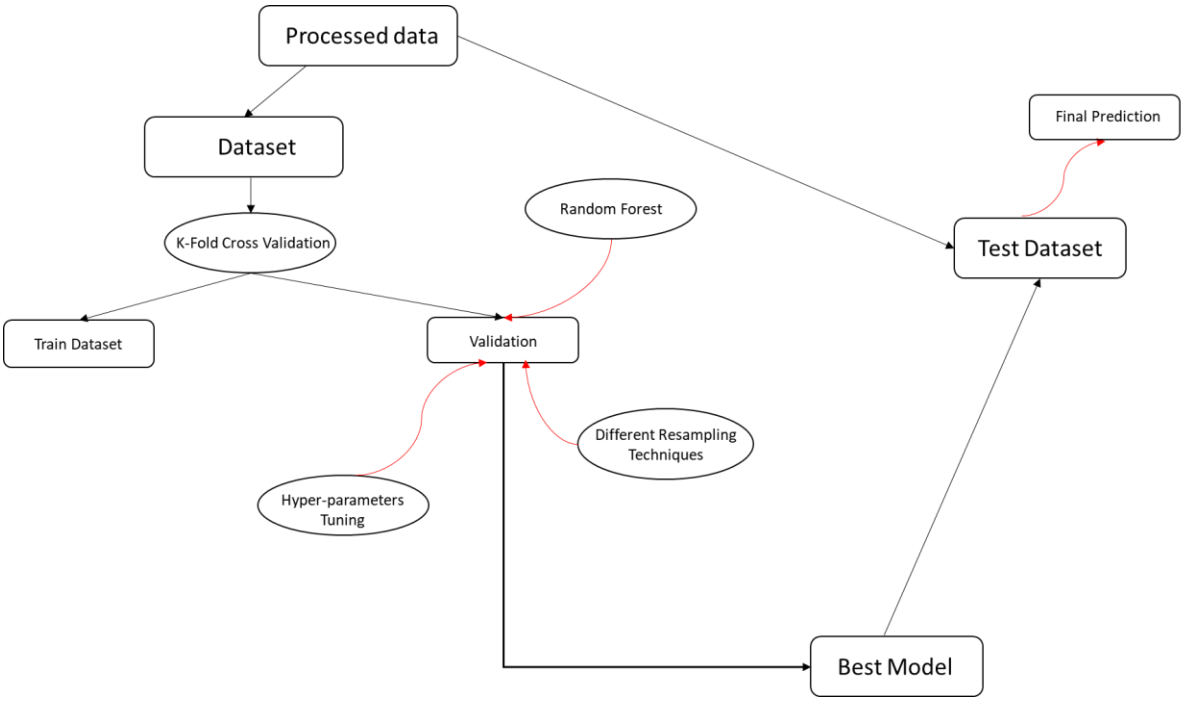


Figure 9: An overview of methodology implemented in this project.

4. Baseline Model

A random forest with default parameters was used to train with the train dataset. Its performance was evaluated by using the model to predict for the test dataset. As illustrated in **Fig.10**, the model obtained weighted average f1-score of 0.97. However, if the individual f1-score for predicting each class was examined, it shows that model can predict class 1, 2,4, and 5 with high accuracy. However, it does relatively poorer job to predict class 0,3, and 6. This is expected because the dataset is highly imbalanced. Therefore, macro average f1-score was chosen as the metric to evaluate the model's performance because it put equal weights for each class.

```
In [6]: print(classification_report(test_pred_baseline,y_test))
              precision    recall  f1-score   support

           0       0.17      1.00      0.29         1
           1       0.96      0.94      0.95       408
           2       0.99      0.95      0.97       622
           3       0.48      0.56      0.52        52
           4       0.99      1.00      0.99       864
           5       0.99      1.00      1.00      1112
           6       0.25      1.00      0.40         3

    accuracy                           0.97      3062
   macro avg       0.69      0.92      0.73      3062
weighted avg       0.98      0.97      0.97      3062
```

Figure 10: The classification report for the baseline random forest.

5. Features Extraction and Features Engineer

Features importance from original dataset was calculated by using Random Forest. As shown in **Fig.11**, "VSH", "PHI", and "W_Tar" were the most three important features in this dataset. Later, two different datasets that derived from the original dataset was used to train and fit two separate baseline random forest model independently. The difference between these two datasets is that one contains the most two important features while the other dataset consists of the three most important features.

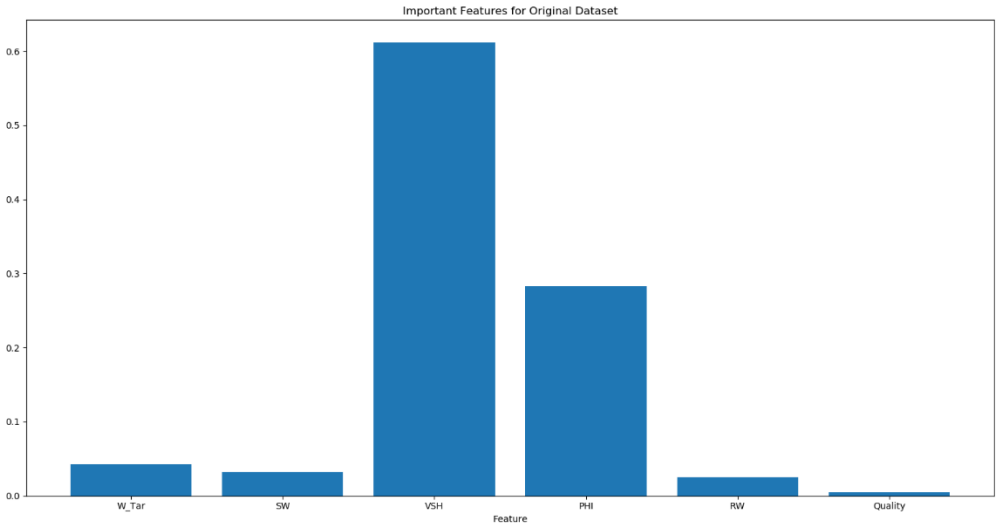


Figure 11: Important features of original dataset.

Subsequently, two engineer features, volume of sand (VS) and hydrocarbon saturation (Shc), were computed by the equation below.

$$VS= 1- VSH$$

$$Shc=1-SW$$

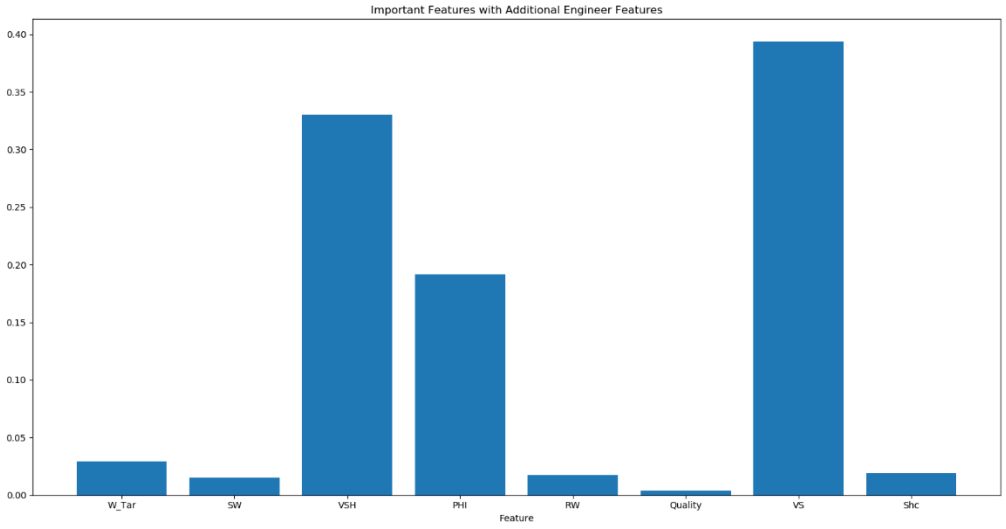


Figure 12: Important features after addition of engineer features.

Feature importance was computed again with this additional engineer features as illustrated in **Fig.12**. Noticed that VS is even more important that PHI and VSH. Shc has about the same importance as SW. A baseline random forest model was trained with this newly dataset. A summary of model with different choice and number of features were shown in **Fig.13**. Because there is not significantly difference in model's performance when fit the model with original dataset or with engineer features; so, the original dataset is used for the rest of this study.

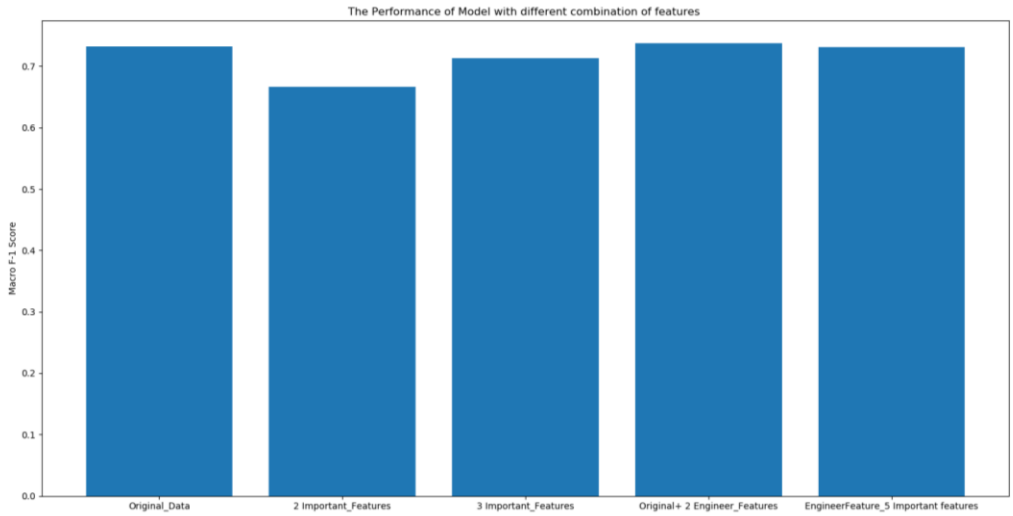


Figure 13: A comparison of model's performance with and without engineer features

6. Handle Imbalanced Dataset – Resampling Techniques

Resampling is a widely adopted technique to handle the imbalanced dataset. It changes the balance of the data by either adding more samples for minority class or reducing samples from majority classes or both. Hence, there are three main methods to resample the data:

- Oversampling: Adds copies of samples from the minority class to obtain a balanced dataset. This is preferred method because the information from the data is not lost.
- Undersampling: Delete copies of samples from the majority class to achieve a balanced dataset. This method should be use with caution because the information form the data may be lost.
- Combination of oversampling and undersampling: Combine the oversampling and undersampling methods may get better results.

In this project, 4 different oversampling methods, one undersampling method and 3 combination method were investigated to see if these techniques can enhance the model performance. RandomOverSampler, SMOTE, ADASYN, and SMOTEENN are oversampling methods. RandomUnderSampler is the only one undersampling technique considered for this project. Two combination methods include SMOTETomek ,SMOTE-RandomUnderSampler and RandomOverSampler-RandomUnderSampler.

- RandomOverSampler: Randomly duplicate records from the minority class.
- SMOTE (Synthetic Minority Ovesampling Technique): Instead of creating copies samples based on those already existed, SMOTE find the K-nearest neighbors of each minority instances and produce a new minority instances in the neighborhood.
- ADASYN (Adaptive Synthetic): Like SMOTE, ADASYN also generates samples of minority class. The only difference between the two methods is that ADASYN focus on generating data for minority class samples that are harder to learn as compared to minority class that is easier to classify.
- SMOTEENN: The combination of SMOTE and Edited Nearest Neighbor. ENN removes any samples whose class labels are different than its neighbors.
- RandomUnderSampler: Randomly delete instances from the majority classes.
- SMOTETomek: Combination of using Tomek to remove instances from the majority classes and using SMOTE to adding more instances for minority classes.
- RandomOverSampler-RandomUnderSampler: Combination of using RandomOverSampler to adding more instances for minority classes while using RandomUnderSampler to reduce instances from majority classes.
- SMOTE-RandomUnderSampler: Combination of using SMOTE to adding more instances for minority classes while using RandomUnderSampler to reduce instances from majority classes.

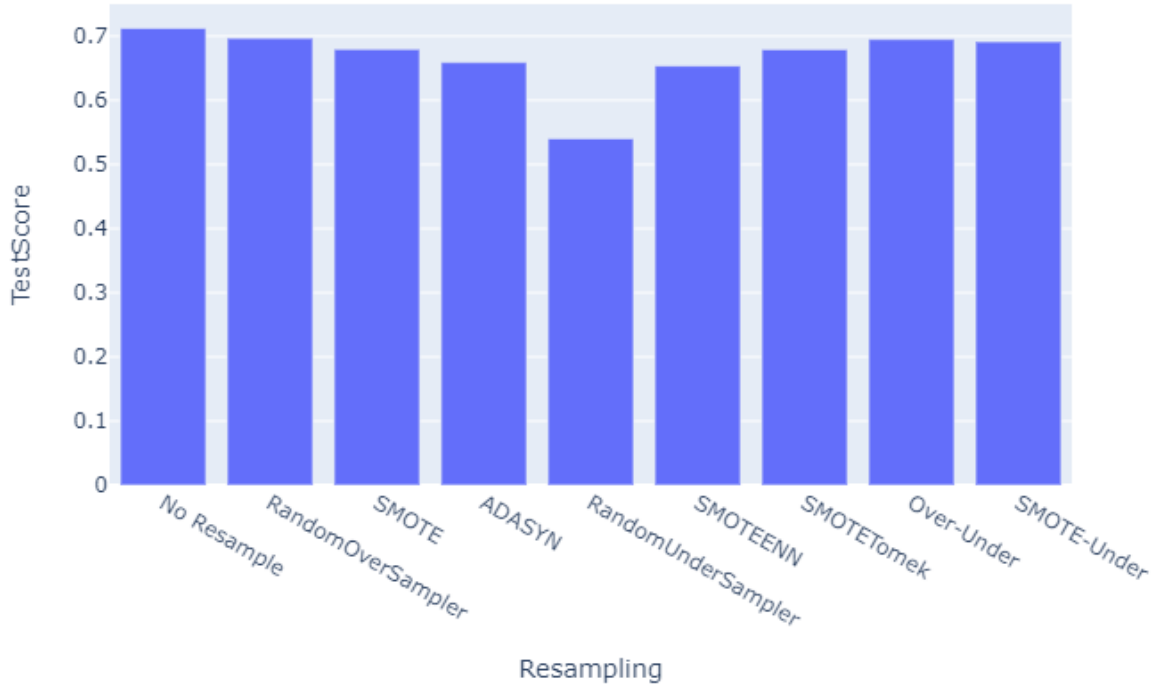
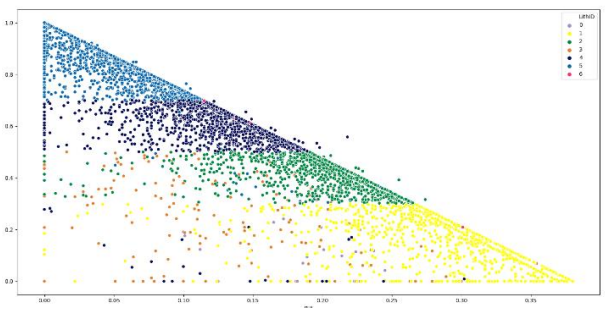


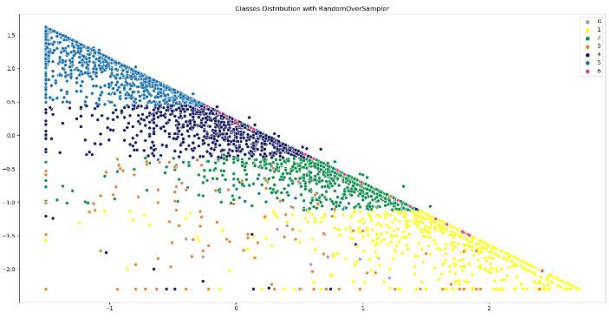
Figure 14: The comparison of model's performance in which different resampling techniques were applied to the original data.

As **Fig.14** illustrated, the model seems to perform the best with the original data rather than with resampling data. RandomUnderSampler performed the worst. **Fig.15** shows how the data looks like after applying resampling technique. These imagines showed that minority classes in this project is very difficult to classify as its instances can spread out from one class cluster to another class cluster. This is understandable as the physical properties of these minority classes are very similar to these majority classes. Therefore, the resampling technique for this dataset does not improve this random forest model.

Original Data

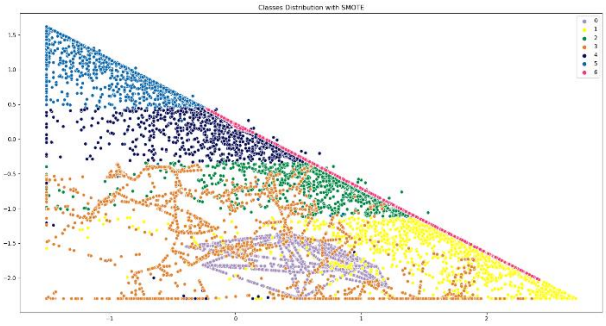


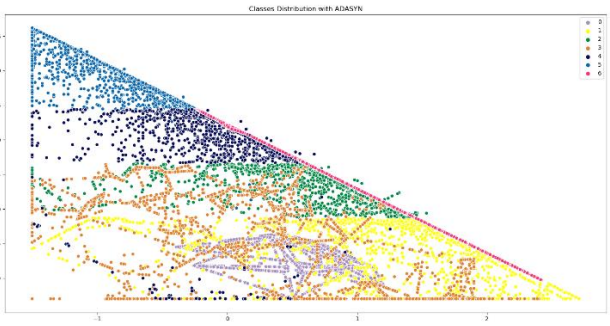
RandomOverSampler



SMOTE

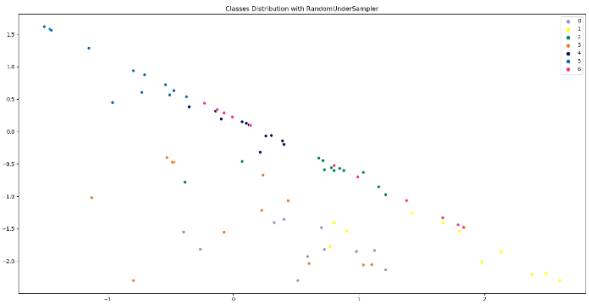
ADASYN

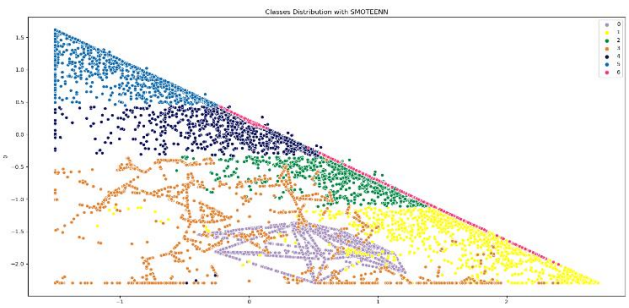




RandomUnderSampler

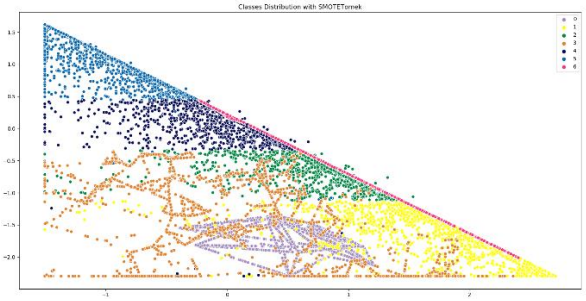
SMOTEENN

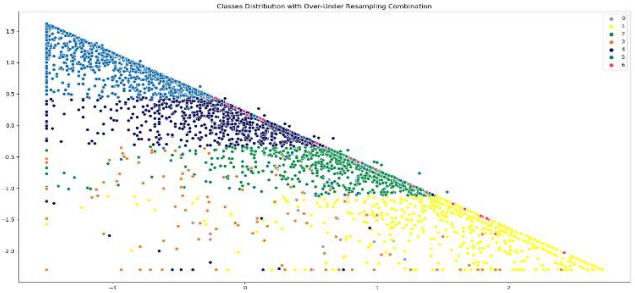




SMOTETomek

RandomOverSampler-RandomUnderSampler





SMOTE-RandomUnderSampler

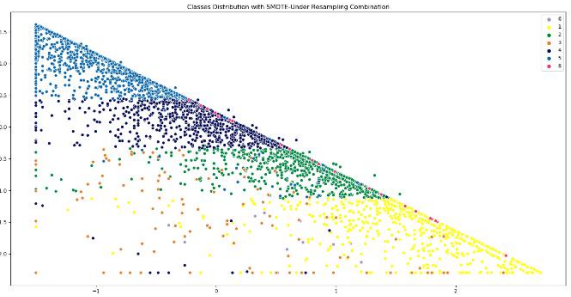


Figure 15: The Effect of Different Resampling Technique to the Original Data

7. Random Forest Hyper-Parameters Tuning

Hyper-parameters were fine-tuned by first apply RandomizedSearchCV to narrow the possible range of values for each random forest's hyper-parameters. Then, GridSearchCV was used to find the optimized model that then verified against the holdout dataset to evaluate its performance. **Table 1** summaries a set of hyper-parameters for this model:

Table 1: A Summary of hyper-parameters used for the final random forest model

Random Forest	N_estimators	Max_Depth	Max_Feature	Min_Sample_Leaf	Min_Sample_Split	Bootstrap	F1- Score for holdout
	557	50	2	1	3	False	0.74

A learning curve of the model with the finetuned hyper-parameters was constructed, as shown in **Fig. 16.** Within probably the first 200 samples of dataset, the F1-score increased steeply from 0 to 0.5. The overall trend is the continuous increase of F1-score metric as more samples were used. Therefore, the model's performance will be improved as more data are collected and used.

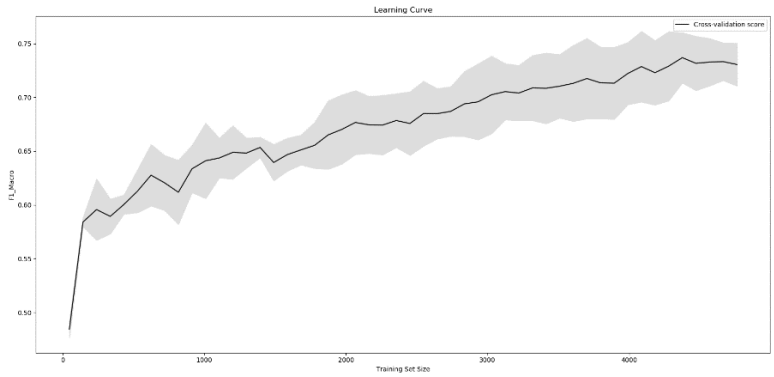


Figure 16: The learning curve of the random forest model with fine-tune hyper-parameters

V. Conclusion

- Various machine learning algorithms were evaluated to determine the potential algorithm for this dataset.
- Ensemble learning, which includes Random Forest, Gradient Boosting and, Stacking outperformed other algorithms.
- Random Forest was chosen for further study. Volume of shale, porosity, volume of bitumen and water saturation were determined to be the most important features according to random forest calculation. However, when using the top 2 and top 3 important features to train the model, the performance of the model is about the same. In addition, two additional engineer features, volume of sand and hydrocarbon saturation, were calculated based on their correlation with volume of shale and water saturation, respectively. The model was then trained with engineer features; nevertheless, the model's performance was about the same as compared to the baseline model.
- Different resampling techniques were applied to the original data to test if it would improve the model. RandomOverSampler, SMOTE, ADASYN, and SMOTEENN were chosen as the

representative for oversampling technique, while RandomUnderSampler was the only choice for undersampling technique. SMOTETomek ,SMOTE-RandomUnderSampler and RandomOverSampler-RandomUnderSampler, were picked as the combination method. Overall, these resampling techniques did not improve the baseline model due to the difficulties to distinguish the minority classes of rocks that have very similar physical properties as compared to majority classes of rock.

- Random Forest was further fined tune with RandomGridSearchCV and GridSearchCV to choose the optimized hyper-parameters. A macro F1-score of 0.74 was obtained with the optimized model.

VI. Recommendation

- More data should be acquired, especially the data for the minority classes, if it is important to predict the minority classes.
- Because the physical properties of minority classes are blended between the majority classes, discuss with experienced geologist to see if more features can be used to describe the minority classes better. Therefore, the model can significantly improve.