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” Datafram

III. Data Exploration

After “Picks” and “Intellog” 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.

```
In [3]: intellog.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 579846 entries, 0 to 579845
Data columns (total 8 columns):
SitID     579846 non-null int64
Depth     579846 non-null float64
LithID    579846 non-null int64
W_Tar     579846 non-null float64
SW        579846 non-null float64
VSH       579846 non-null float64
PHI       579846 non-null float64
RW        579846 non-null object
dtypes: float64(5), int64(2), object(1)
memory usage: 35.4+ MB
```

```
In [4]: picks.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30702 entries, 0 to 30701
Data columns (total 4 columns):
SitID      30702 non-null int64
HorID      30702 non-null int64
Pick       30702 non-null object
Quality    30702 non-null int64
dtypes: int64(3), object(1)
memory usage: 959.5+ KB
```

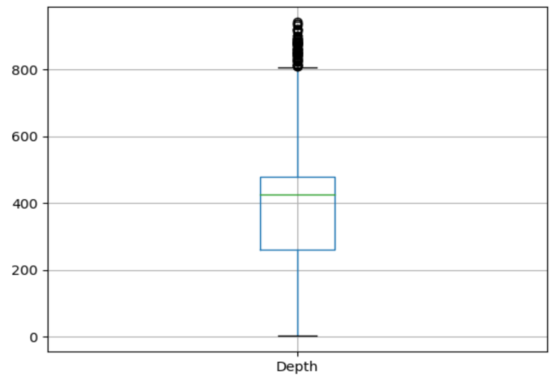
As shown, there were 579,846 values in each column of “Intellog” dataframe. No NaN values were present in this dataframe. Columns “SitID”, “LithID” has data type as int64. Columns “Depth”, “W_Tar”, “SW”,”VSH”, and “PHI” were in float64 format. However, column “RW” are in non-null object. So, this column “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.

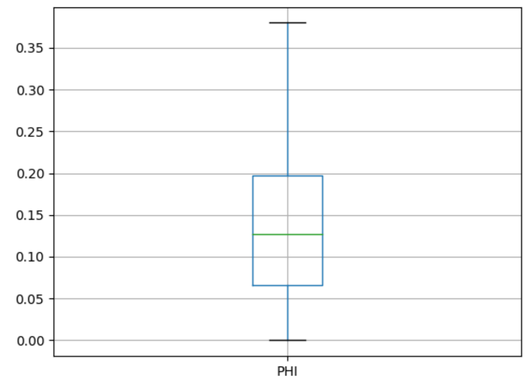
Next, the two dataframes were merged by columns “SitID” and “Depth”. As shown, there were 10,318 values in each column, except column “RW” which only had 10,204 values. Therefore, the NaN can be dropped from dataframe.

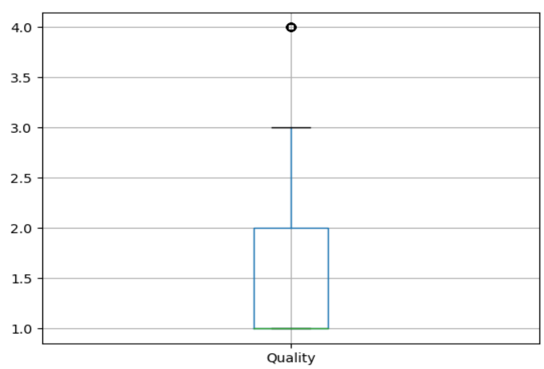
```
In [7]: main_file.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10318 entries, 0 to 10317
Data columns (total 10 columns):
SitID      10318 non-null int64
Depth      10318 non-null float64
LithID     10318 non-null int64
W_Tar      10318 non-null float64
SW         10318 non-null float64
VSH        10318 non-null float64
PHI        10318 non-null float64
RW         10204 non-null float64
HorID      10318 non-null int64
Quality    10318 non-null int64
dtypes: float64(6), int64(4)
memory usage: 886.7 KB
```

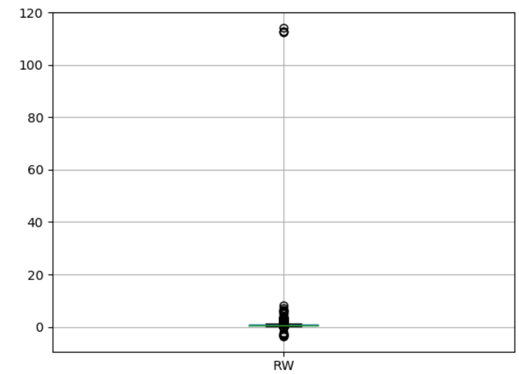
The data distribution can also be viewed by using df.describe(). However, box plots will be more efficient to visualize its data distribution. As illustrated, majority of features have outliers in its data except for “PHI” column. Further investigation on these outliers should be carried out to understand why these outliers existed. For this project, these outliers were left alone as it does not affect the model performance as shown later.

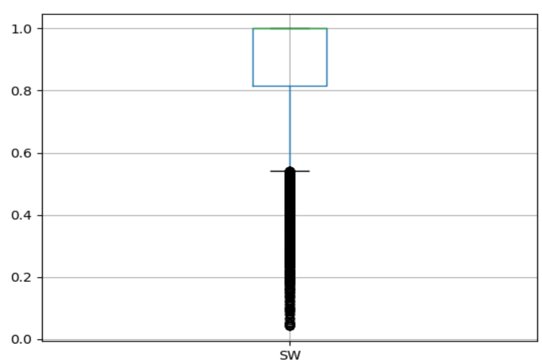
```
In [15]: main_file.describe()
Out[15]:
              SitID         Depth        LithID         W_Tar            SW  \
count  10318.000000  10318.000000  10318.000000  10318.000000  10318.000000
mean  121259.241132    382.783003      3.568133     -0.002355      0.878563
std    19171.458013    156.116885      1.479058      0.123287      0.211620
min   102496.000000      2.000000      0.000000     -1.117000      0.041000
25%   108617.250000    260.500000      2.000000      0.000000      0.817000
50%   113215.500000    425.500000      4.000000      0.000000      1.000000
75%   125118.000000    480.000000      5.000000      0.010000      1.000000
max   184130.000000    942.000000      6.000000      0.173000      1.000000

                VSH           PHI            RW         HorID       Quality
count  10318.000000  10318.000000  10204.000000  10318.000000  10318.000000
mean       0.587704      0.135197      0.705499  10014.634619      1.672514
std        0.255268      0.090185      2.248045   2783.586918      0.823260
min        0.000000      0.000000     -3.560000   2000.000000      1.000000
25%        0.414000      0.066000      0.511000   9000.000000      1.000000
50%        0.606000      0.127000      0.629000  10000.000000      1.000000
```

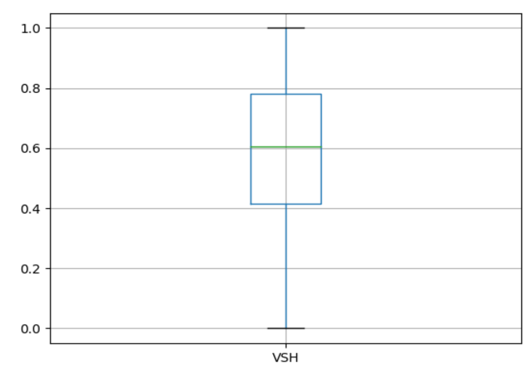

0
200
400
600
800
Depth

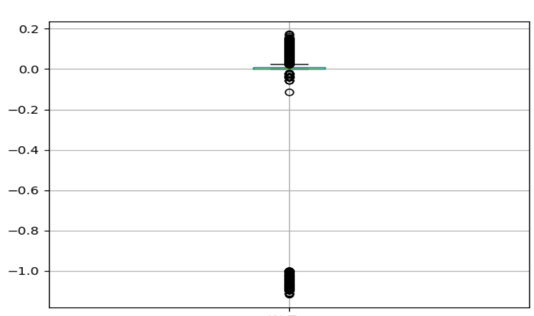

0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
PHI


1.0
1.5
2.0
2.5
3.0
3.5
4.0
Quality


0
20
40
60
80
100
120
RW


0.0
0.2
0.4
0.6
0.8
1.0
SW


0.0
0.2
0.4
0.6
0.8
1.0
VSH

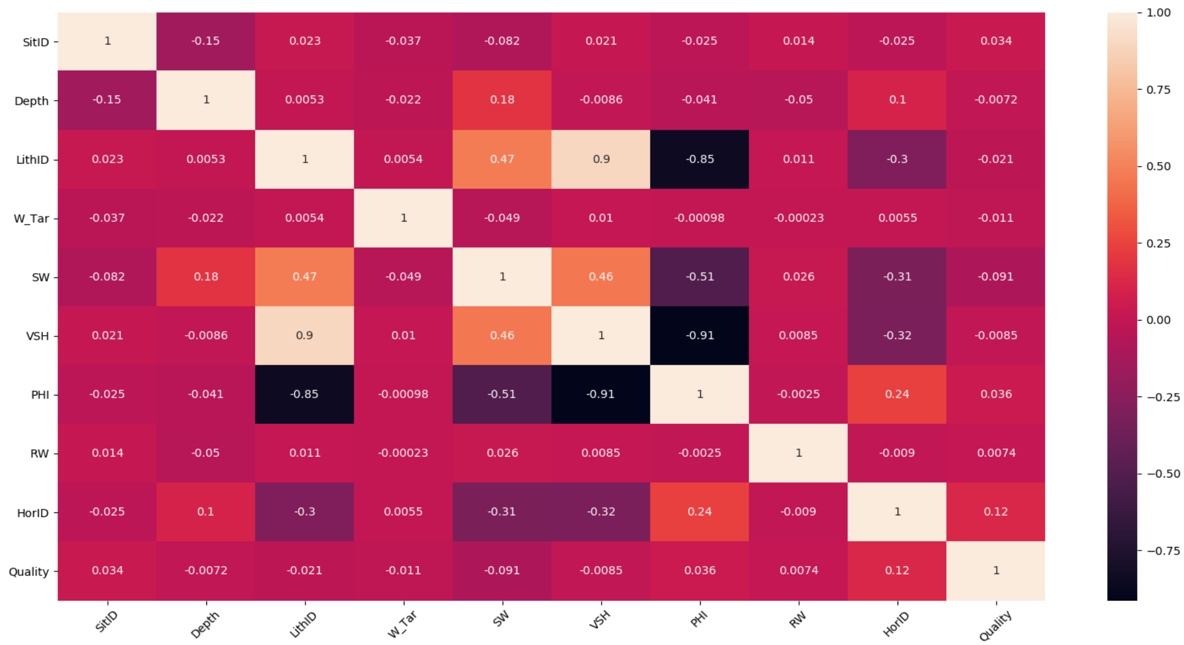

0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
W_Tar

Lithofacies classification is the target of this project. It is necessary to see how different facies distributed within this dataset. As shown from the figure below, the proportions of classes 1,2,4, and 5 are about 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.

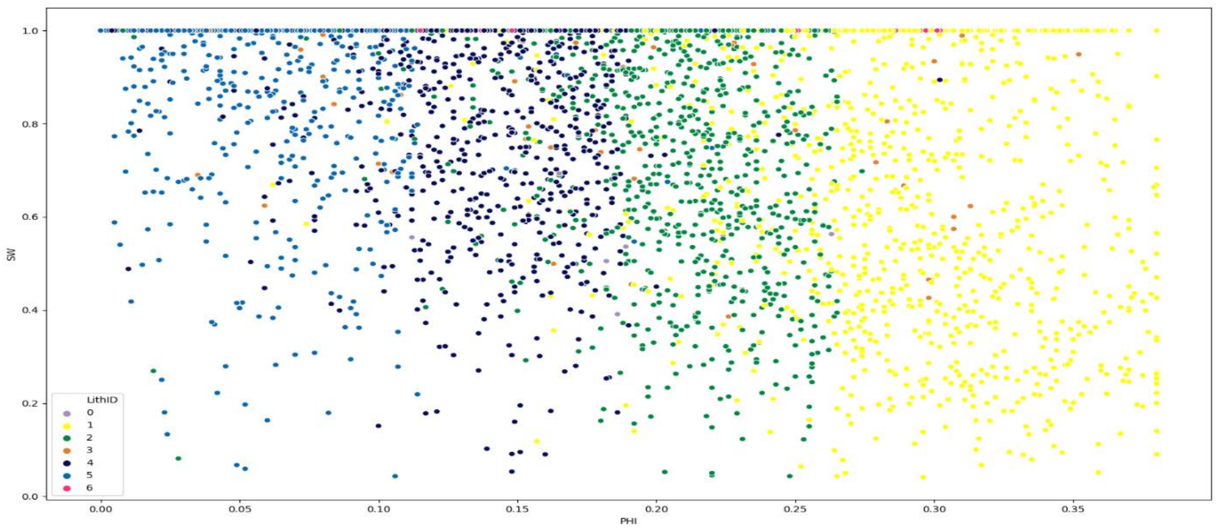
```
Class=2, Count=1999, Percentage= 19.59%
Class=4, Count=2898, Percentage= 28.40%
Class=1, Count=1321, Percentage= 12.95%
Class=5, Count=3726, Percentage= 36.52%
Class=3, Count=200, Percentage= 1.96%
Class=0, Count=20, Percentage= 0.20%
Class=6, Count=40, Percentage= 0.39%
```

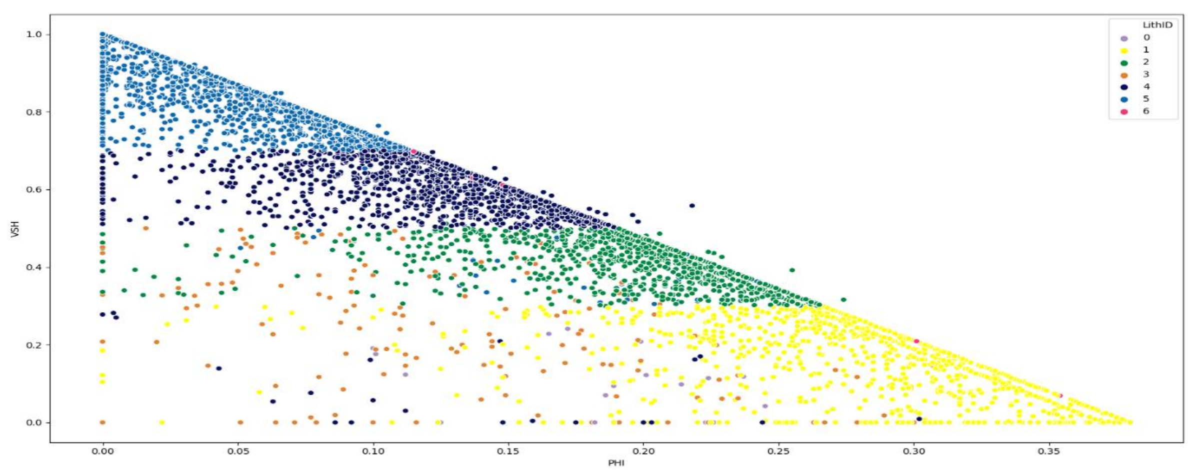
A correlation heat map is constructed to examine the correlation between features. Several correlations were observed:

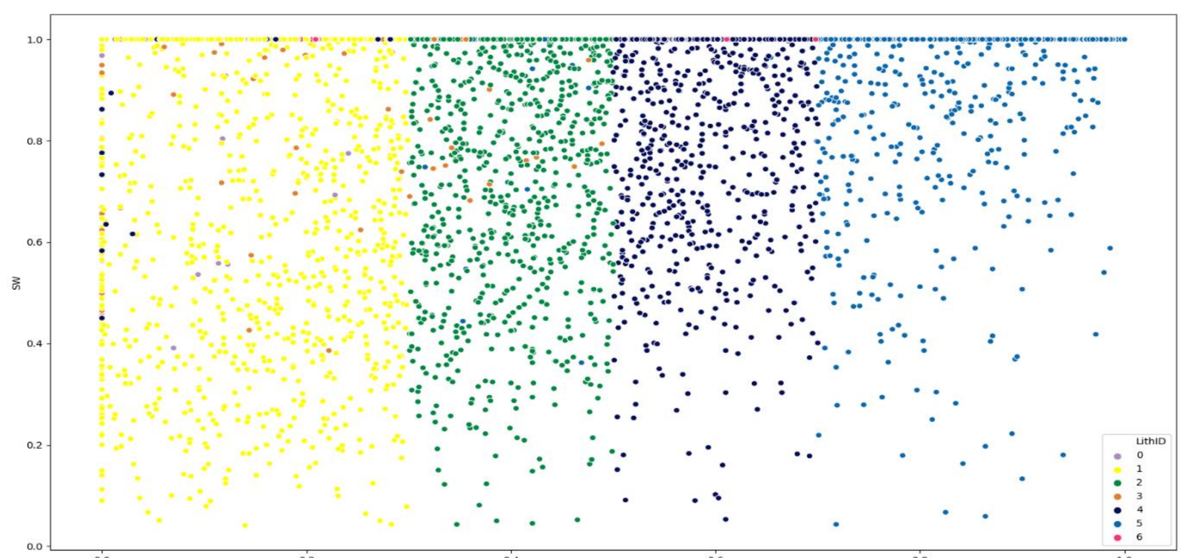
- Strong correlation between volume of shale and lithology
- Strong negative correlation between volume of shale and porosity
- Strong correlation between porosity and lithology
- Good correlation between water saturation and volume of shale
- Good negative correlation between porosity and water saturation



Because there is strong correlation between volume of shale, porosity, and water saturation to lithology classes, I am wondering how lithology classes distribution look like when these parameters are plotted against each other.


SW
PHI
LithID
0
1
2
3
4
5
6
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35


VSH
PHI
LithID
0
1
2
3
4
5
6
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35


SW
VSH
LithID
0
1
2
3
4
5
6
0.0
0.2
0.4
0.6
0.8
1.0

From these plots above, lithology classes 1,2,4, and 5 are distinct clusters that the machine learning model can classify them correctly. Nevertheless, lithology class 0,3 and 6 are mixed into other classes as expected because the rock that belong to these classes have very similar properties with 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.