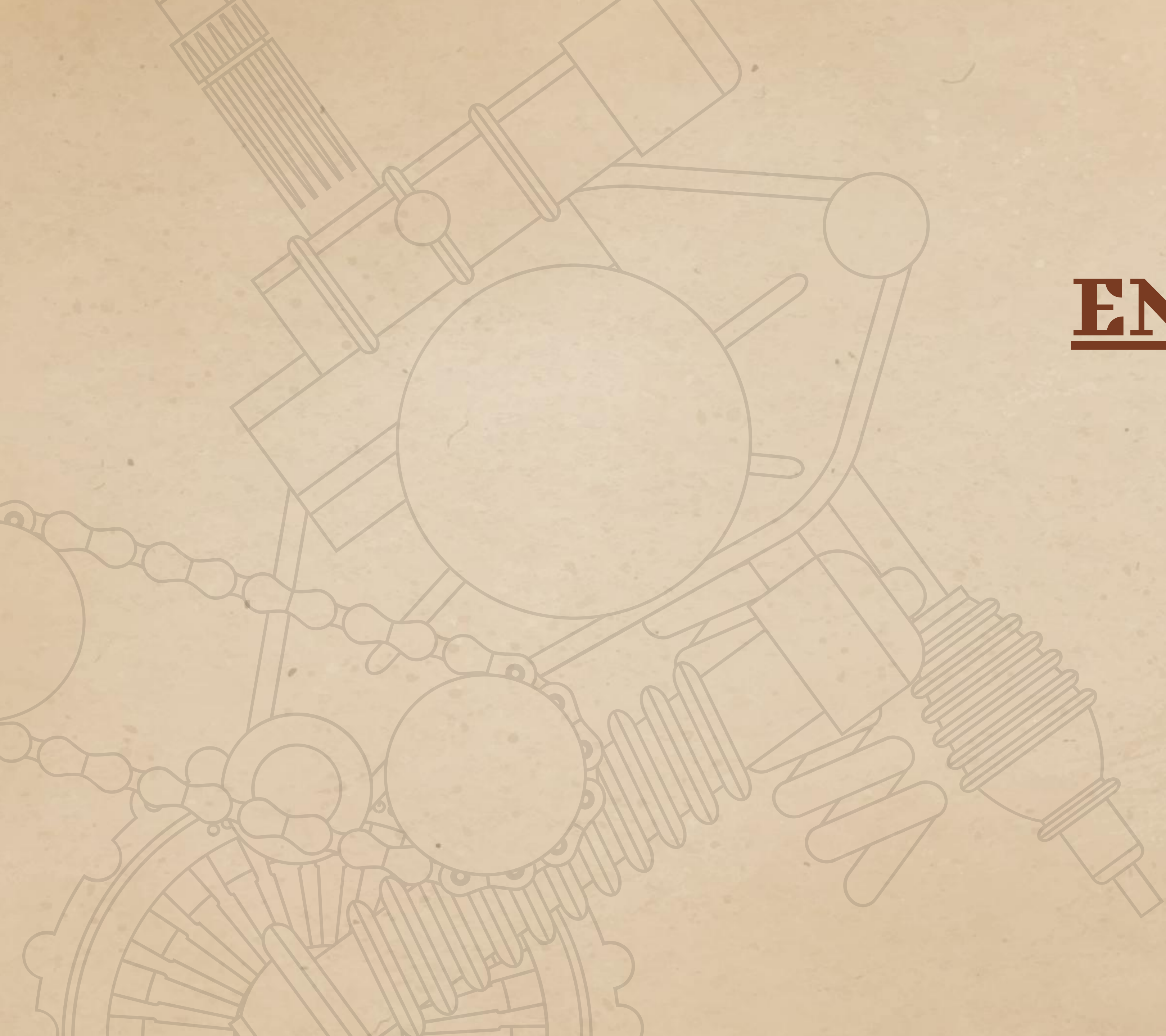


ENGINE HEALTH PREDICTION

- PRESENTED BY : Parthasarathi

TABLE OF CONTENTS

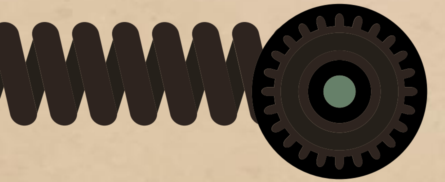
Problem Statement

★ In order to maintain optimal engine performance, prevent potential engine failures, and enhance overall efficiency.Additionally, a key aspect of engine optimization involves reducing the environmental impact associated with emissions.

★ By prioritizing the prevention of engine failures, improving performance, and minimizing the environmental impact of emissions, we can achieve a harmonious balance between efficient operations, sustainable practices, and environmental supervision

★ Develop machine learning model to accurately predict the health status of engines based on the real-time analysis of multiple engine parameters, including

❈ Engine RPM ❈ Lubricating Oil Pressure ❈ Fuel Pressure ❈ Coolant Pressure
❈ Lubricating Oil Temperature ❈ Coolant Temperature.



Data Definition :

📰 Engine rpm : RPM stands for "Revolutions Per Minute" and refers to the rotational speed of engine (Numeric Variable).
📰 Lub oil pressure : It refers to pressure exerted by the lubricating oil within an engine's lubrication system (Numeric Variable,Measured in Pounds per square inch (PSI)).

📰 Fuel pressure : It represents the force at which the fuel is delivered to the engine's combustion chambers (Numeric Variable,Measured in Pounds per square inch (PSI)).

📰 Coolant pressure : It represents the force at which the coolant circulates through the system, helping to regulate and maintain the engine's optimal operating temperature (Numeric Variable,Measured in Pounds per square inch (PSI)).

📰 Lub oil temp : It refers to the temperature of the engine's lubricating oil within the lubrication system (Numeric Variable,Measured in Degrees Fahrenheit (°F)).

📰 Coolant temp : It refers to the temperature of the coolant within the cooling system of an engine (Numeric Variable,Measured in Degrees Fahrenheit (°F)).

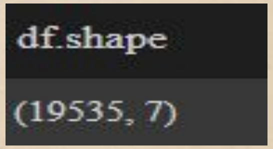
📰 Engine Condition : It referes to condition of the engine Healthy Engine or UnHealthy Engine. it varies based on Engine RPM , Lubricating Oil Pressure, Fuel Pressure, Coolant Pressure, Lubricating Oil Temperature, Coolant Temperature (Target Variable : Un-Healthy Engine = 0 and Healthy Engine = 1).

Data & Feature :

Link to Dataset :

https://www.kaggle.com/datasets/parvmodi/automotive-vehicles-engine-health-dataset

About Dataset :



```
df.shape
(19535, 7)
```

Shape of DataFrame

★ Dataset consist of 19535 records and 7 rows. In data set “Engine Condition” is target variable other columns are independent variable

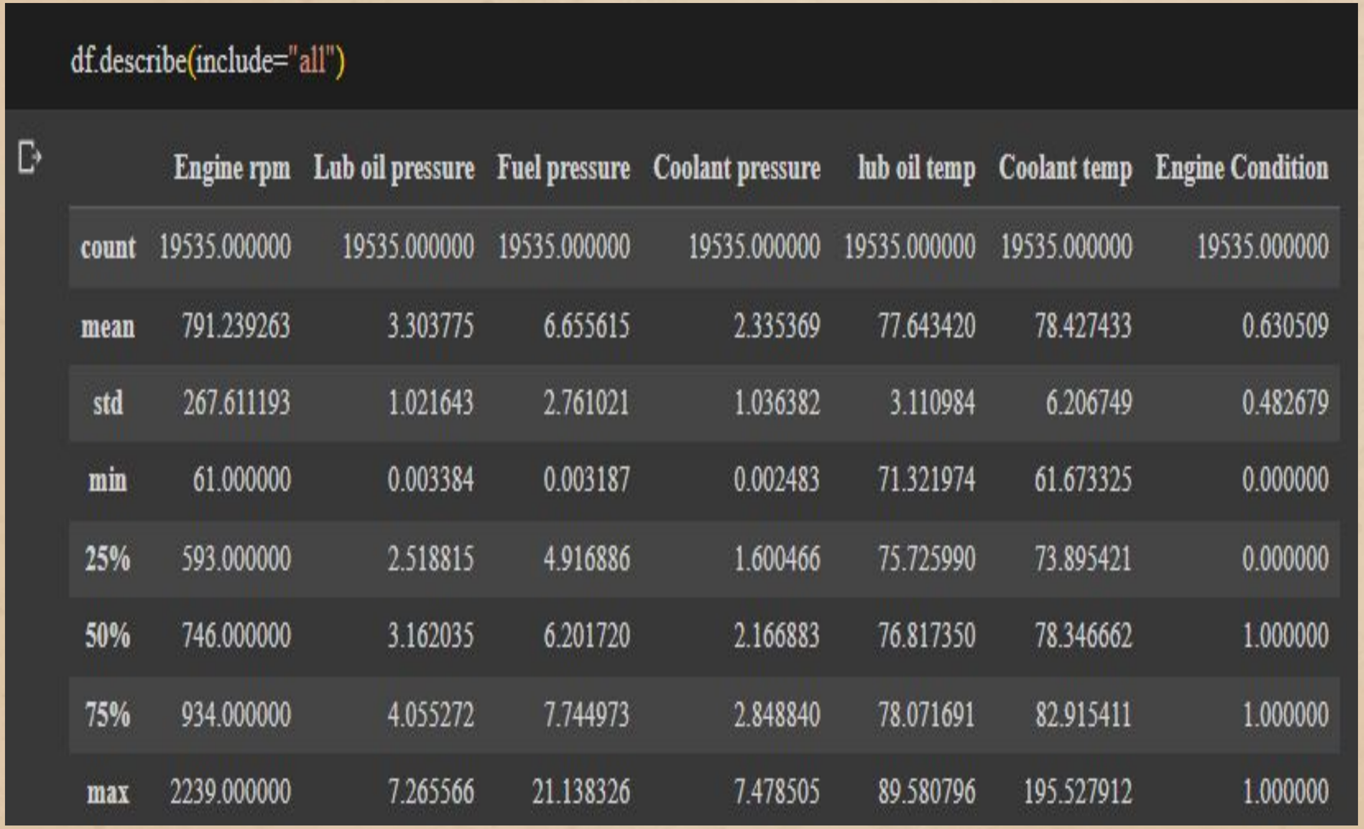
★ In this data over target variable is categorical of binary classification (0,1).

★ So we are going to build model based on some of Classification type machine learning algorithms are Logistic Regression, Decision Tree Classifier, Random Forest Classifier, K-Nearest Neighbor, Support Vector Machine, Extreme Gradient Boosting,

EDA

Identifying the data types of each column, Detecting and handling missing values if any, Analyzing the distribution of values in each column, Visualizing the relationship between variables

Descriptive statistics of a DataFrame



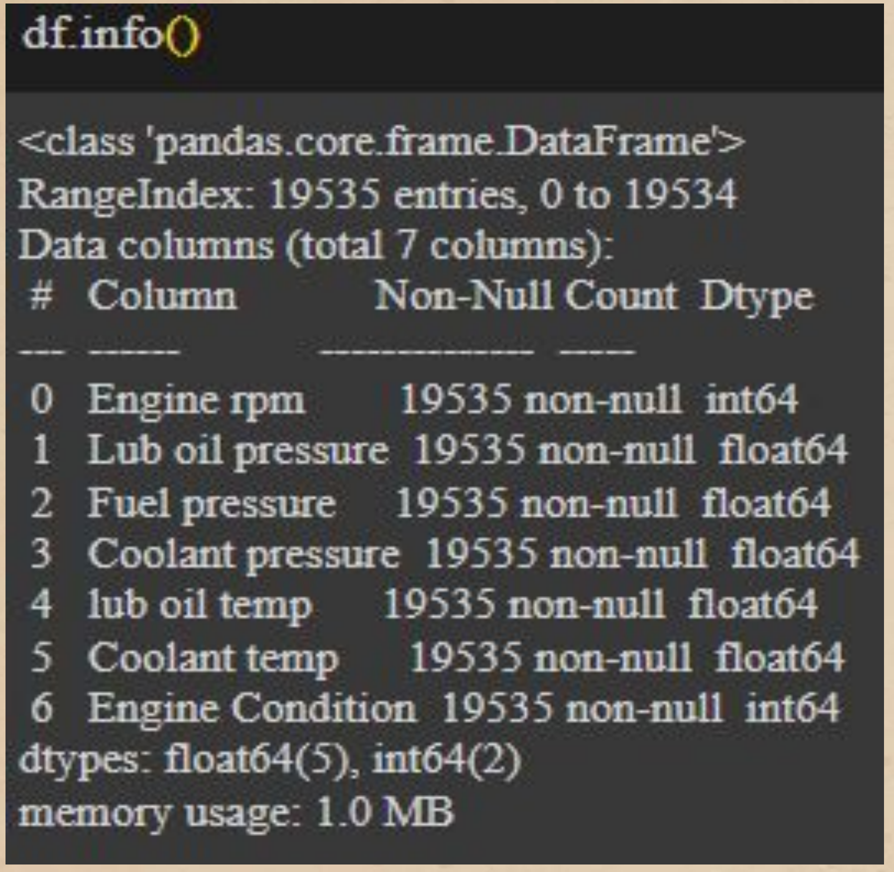
```
df.describe(include="all")
```

```
       Engine rpm  Lub oil pressure  Fuel pressure  Coolant pressure  lub oil temp  Coolant temp  Engine Condition
count  19535.000000      19535.000000   19535.000000      19535.000000  19535.000000  19535.000000      19535.000000
mean     791.239263          3.303775       6.655615          2.335369     77.643420     78.427433          0.630509
std      267.611193          1.021643       2.761021          1.036382      3.110984      6.206749          0.482679
min       61.000000          0.003384       0.003187          0.002483     71.321974     61.673325          0.000000
25%      593.000000          2.518815       4.916886          1.600466     75.725990     73.895421          0.000000
50%      746.000000          3.162035       6.201720          2.166883     76.817350     78.346662          1.000000
75%      934.000000          4.055272       7.744973          2.848840     78.071691     82.915411          1.000000
max     2239.000000          7.265566      21.138326          7.478505     89.580796    195.527912          1.000000
```

- ★ Count: The number of non-null values in the column.
- ★ Mean: The average value of the column.
- ★ Standard Deviation: The measure of the variability or spread of the values in the column.
- ★ Minimum: The smallest value in the column.
- ★ 25th Percentile (Q1): The value below which 25% of the data falls.
- ★ 50th Percentile (Q2 or Median): The value below which 50% of the data falls.
- ★ 75th Percentile (Q3): The value below which 75% of the data falls.
- ★ Maximum: The largest value in the column.

Concise summary :

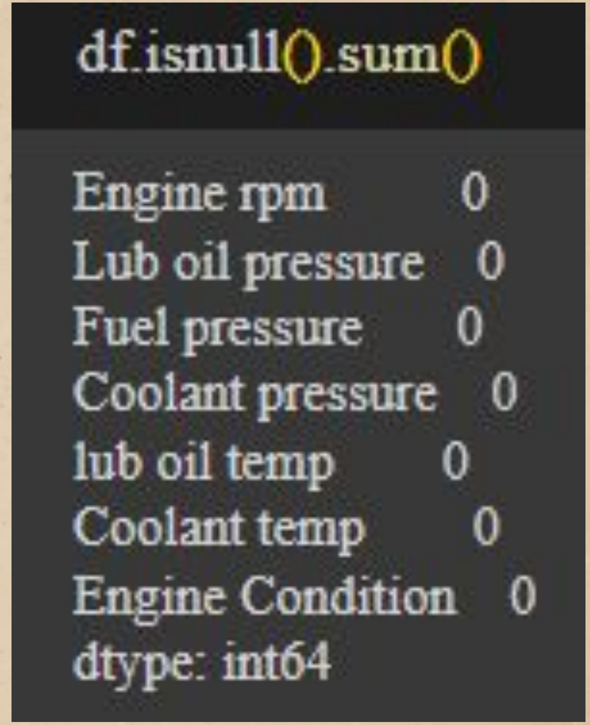
- ★ In the DataFrame, there are 19535 non-null values for all fields.
- ★ The fields "Engine RPM" and "Engine Condition" have integer data types, while the other fields are of float data type.



```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19535 entries, 0 to 19534
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Engine rpm        19535 non-null  int64
 1   Lub oil pressure  19535 non-null  float64
 2   Fuel pressure     19535 non-null  float64
 3   Coolant pressure  19535 non-null  float64
 4   lub oil temp      19535 non-null  float64
 5   Coolant temp      19535 non-null  float64
 6   Engine Condition  19535 non-null  int64
dtypes: float64(5), int64(2)
memory usage: 1.0 MB
```

Check For Tot.No.Of. Null value

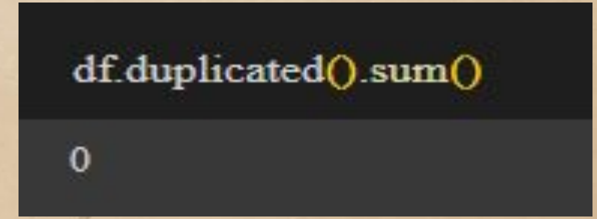
- ★ The DataFrame has no missing values in any of its columns.



```
df.isnull().sum()

Engine rpm          0
Lub oil pressure    0
Fuel pressure       0
Coolant pressure    0
lub oil temp        0
Coolant temp        0
Engine Condition    0
dtype: int64
```

Check for Duplicates :

- ★ The DataFrame contains only unique records; there are no duplicated rows



```
df.duplicated().sum()

0
```

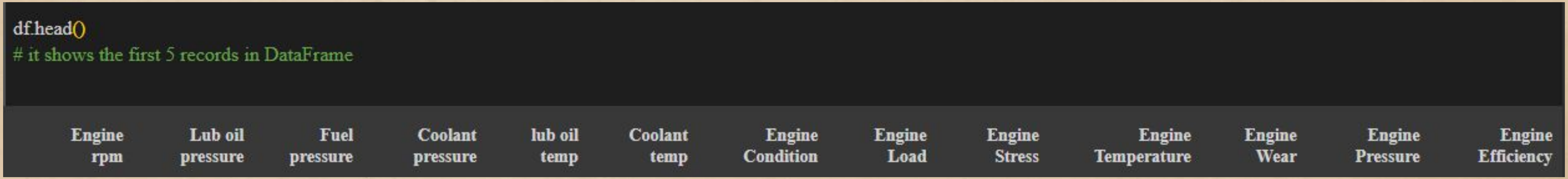
Feautre Extraction :

★ Feature extraction, It involves selecting or extracting relevant characteristics or properties from the data that are expected to be most predictive or relevant for the task at hand.

★ After extracting the relevant and important features to handle the dataset for predicting the Engine Condition, we are adding the following features to the DataFrame:

❄ Engine Load ❄ Engine Stress ❄ Engine Temperature
❄ Engine Wear ❄ Engine Pressure ❄ Engine Efficiency.

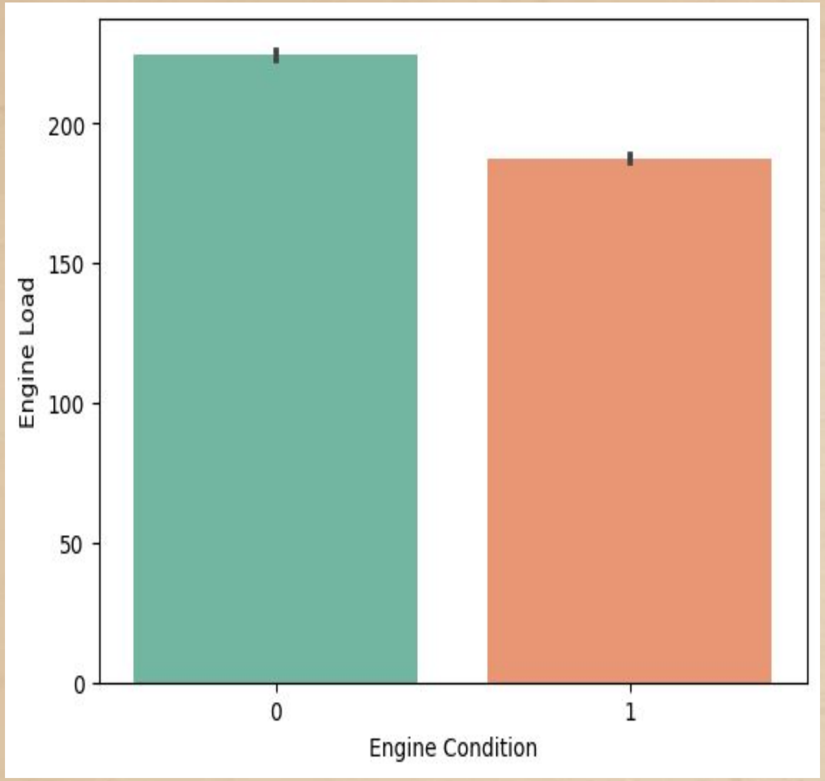
These features will be incorporated into the DataFrame for further analysis and modeling tasks related to predicting the Engine Condition.

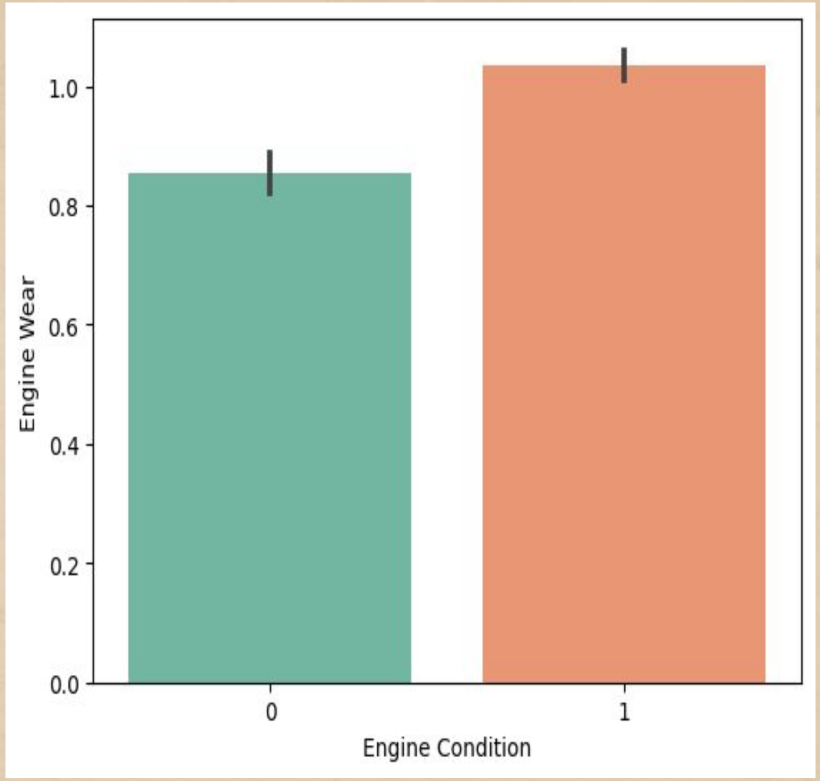


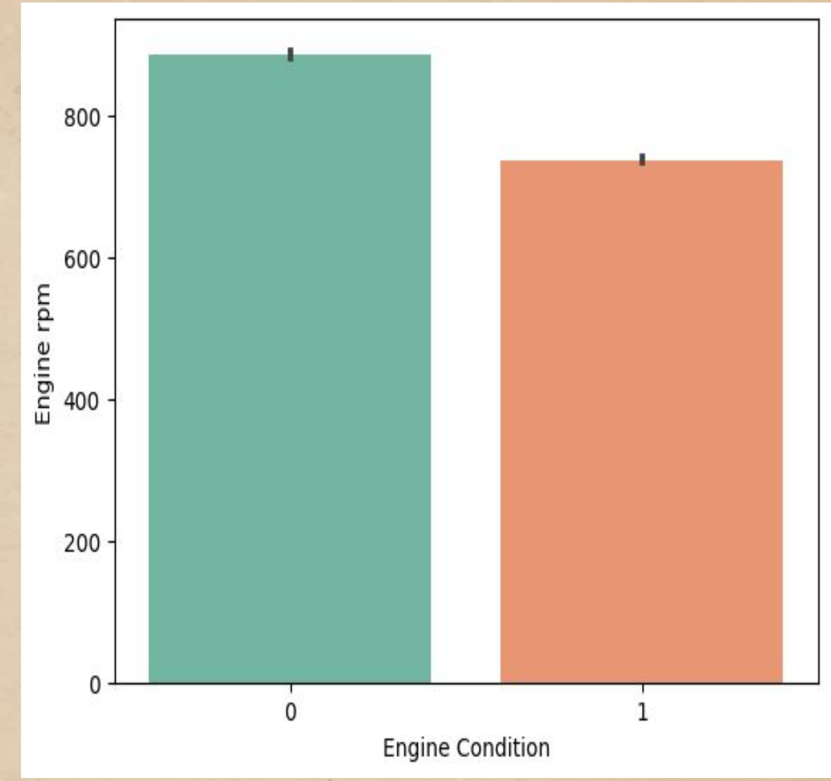
```python
df.head()
# it shows the first 5 records in DataFrame
```

	Engine rpm	Lub oil pressure	Fuel pressure	Coolant pressure	lub oil temp	Coolant temp	Engine Condition	Engine Load	Engine Stress	Engine Temperature	Engine Wear	Engine Pressure	Engine Efficiency

Data visualization :



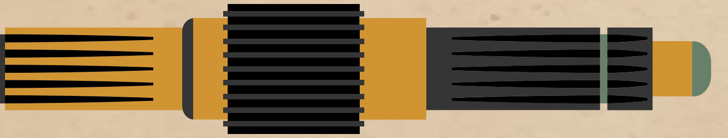




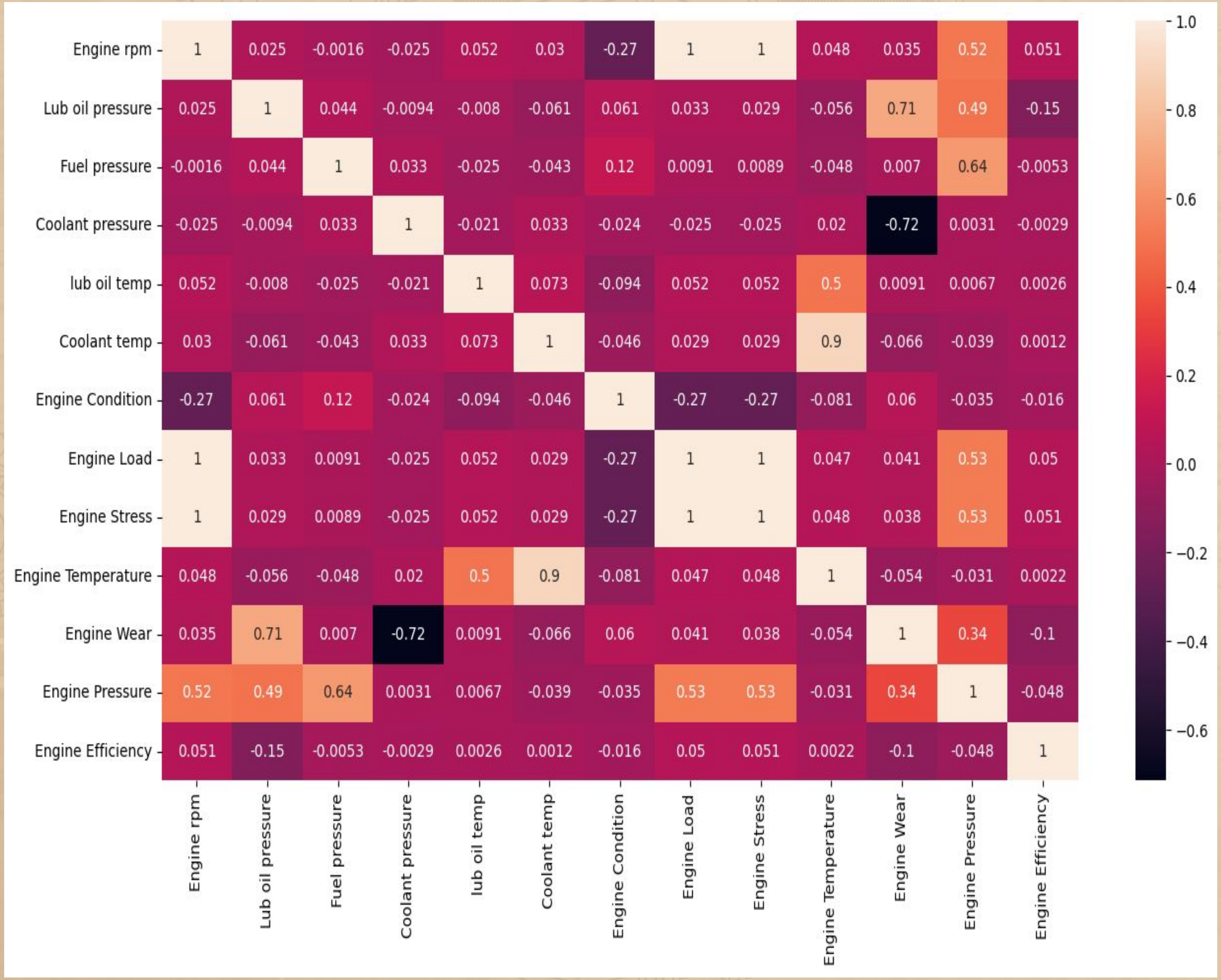
Engine Load
Vs
Engine Condition

Engine Wear
Vs
Engine Condition

Engine RPM
Vs
Engine Condition

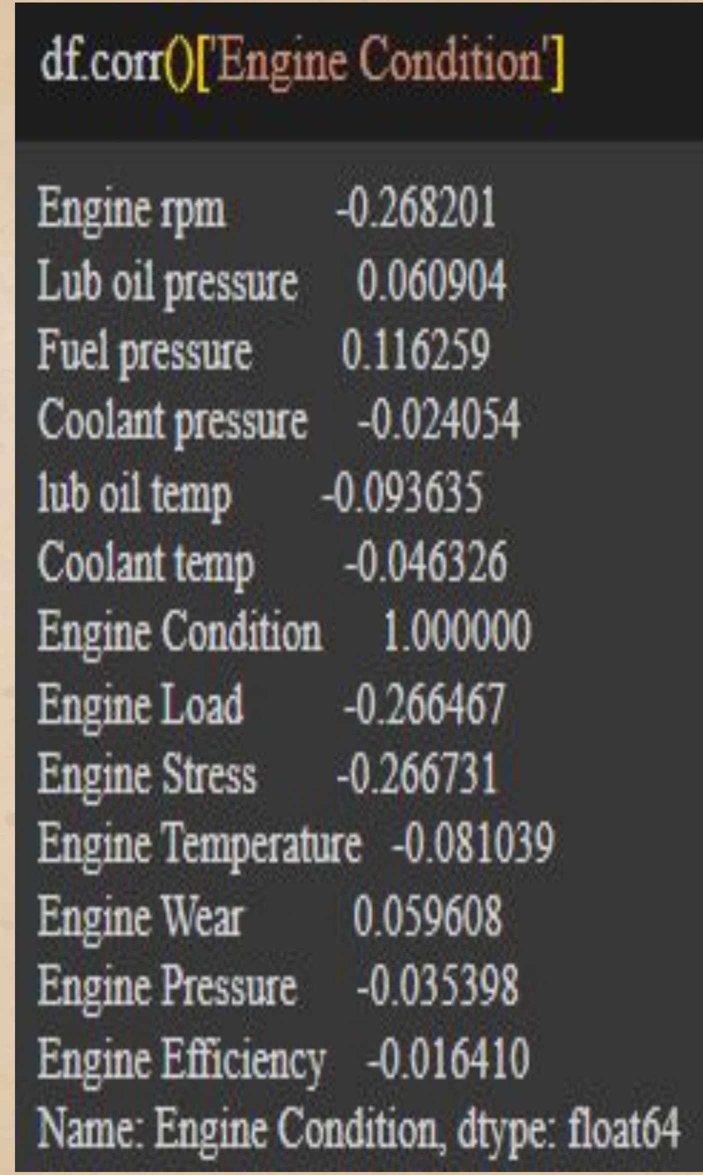


Heatmap :



	Engine rpm	Lub oil pressure	Fuel pressure	Coolant pressure	lub oil temp	Coolant temp	Engine Condition	Engine Load	Engine Stress	Engine Temperature	Engine Wear	Engine Pressure	Engine Efficiency
Engine rpm	1	0.025	-0.0016	-0.025	0.052	0.03	-0.27	1	1	0.048	0.035	0.52	0.051
Lub oil pressure	0.025	1	0.044	-0.0094	-0.008	-0.061	0.061	0.033	0.029	-0.056	0.71	0.49	-0.15
Fuel pressure	-0.0016	0.044	1	0.033	-0.025	-0.043	0.12	0.0091	0.0089	-0.048	0.007	0.64	-0.0053
Coolant pressure	-0.025	-0.0094	0.033	1	-0.021	0.033	-0.024	-0.025	-0.025	0.02	-0.72	0.0031	-0.0029
lub oil temp	0.052	-0.008	-0.025	-0.021	1	0.073	-0.094	0.052	0.052	0.5	0.0091	0.0067	0.0026
Coolant temp	0.03	-0.061	-0.043	0.033	0.073	1	-0.046	0.029	0.029	0.9	-0.066	-0.039	0.0012
Engine Condition	-0.27	0.061	0.12	-0.024	-0.094	-0.046	1	-0.27	-0.27	-0.081	0.06	-0.035	-0.016
Engine Load	1	0.033	0.0091	-0.025	0.052	0.029	-0.27	1	1	0.047	0.041	0.53	0.05
Engine Stress	1	0.029	0.0089	-0.025	0.052	0.029	-0.27	1	1	0.048	0.038	0.53	0.051
Engine Temperature	0.048	-0.056	-0.048	0.02	0.5	0.9	-0.081	0.047	0.048	1	-0.054	-0.031	0.0022
Engine Wear	0.035	0.71	0.007	-0.72	0.0091	-0.066	0.06	0.041	0.038	-0.054	1	0.34	-0.1
Engine Pressure	0.52	0.49	0.64	0.0031	0.0067	-0.039	-0.035	0.53	0.53	-0.031	0.34	1	-0.048
Engine Efficiency	0.051	-0.15	-0.0053	-0.0029	0.0026	0.0012	-0.016	0.05	0.051	0.0022	-0.1	-0.048	1

Correlation Values :



```
df.corr()['Engine Condition']

Engine rpm           -0.268201
Lub oil pressure      0.060904
Fuel pressure         0.116259
Coolant pressure     -0.024054
lub oil temp         -0.093635
Coolant temp         -0.046326
Engine Condition      1.000000
Engine Load          -0.266467
Engine Stress        -0.266731
Engine Temperature   -0.081039
Engine Wear           0.059608
Engine Pressure      -0.035398
Engine Efficiency    -0.016410
Name: Engine Condition, dtype: float64
```

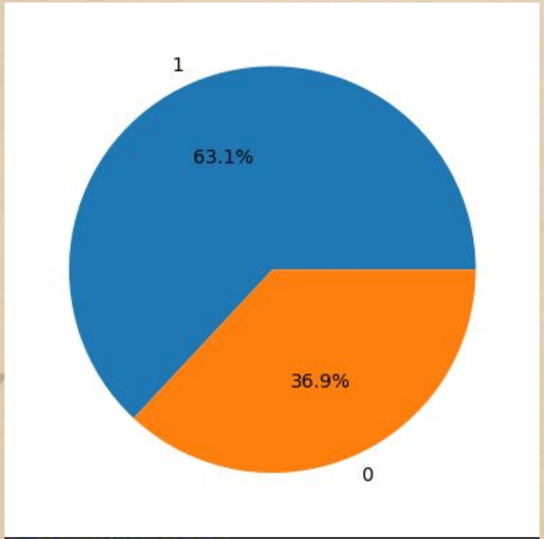
Observed From the Heatmap & Correlation values

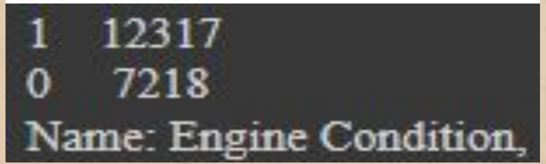
- ★ Engine rpm" has a moderate negative correlation with engine condition, meaning higher engine rpm is associated with lower engine condition.
- ★ "Lub oil pressure" has a weak positive correlation with engine condition, suggesting that higher lub oil pressure is slightly associated with better engine condition.
- ★ "Fuel pressure" has a weak positive correlation with engine condition, indicating that higher fuel pressure may be related to better engine condition, although the correlation is not strong.
- ★ "Coolant pressure" has a weak negative correlation with engine condition, suggesting that higher coolant pressure may be associated with slightly lower engine condition.
- ★ "Lub oil temp" has a weak negative correlation with engine condition, meaning that higher lub oil temperature may be related to slightly lower engine condition.
- ★ "Coolant temp" has a weak negative correlation with engine condition, indicating that higher coolant temperature may be associated with slightly lower engine condition.
- ★ "Engine Load" has a moderate negative correlation with engine condition, indicating that higher engine load is associated with lower engine condition.
- ★ "Engine Stress" has a moderate negative correlation with engine condition, suggesting that higher engine stress tends to be associated with lower engine condition.
- ★ "Engine Temperature" has a weak negative correlation with engine condition, meaning that higher engine temperature may be associated with slightly lower engine condition.
- ★ "Engine Wear" has a weak positive correlation with engine condition, suggesting that higher engine wear may be related to better engine condition, although the correlation is not strong.
- ★ "Engine Pressure" has a weak negative correlation with engine condition, indicating that higher engine pressure may be associated with slightly lower engine condition.
- ★ "Engine Efficiency" has a very weak negative correlation with engine condition, meaning that engine efficiency has almost no linear relationship with engine condition.

Class Balancing :

Class balancing is important in machine learning when dealing with imbalanced datasets. To improve Accuracy and Performance, Preventing Information Loss, Improved Generalization,Better Evaluation Metrics.

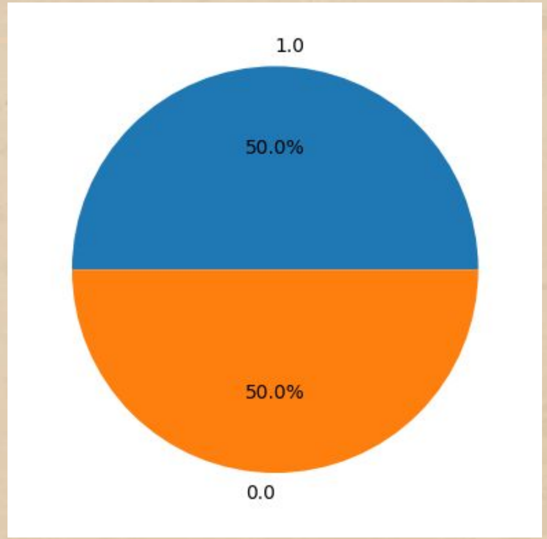
BEFORE BALANCING

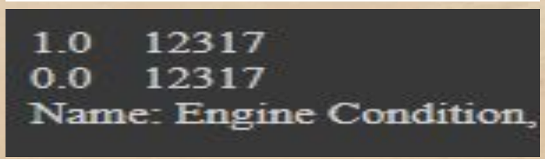




```
1    12317
0     7218
Name: Engine Condition,
```

AFTER BALANCING

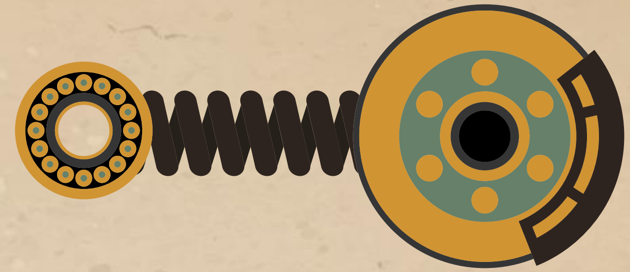




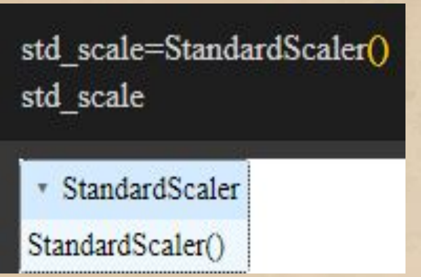
```
1.0    12317
0.0    12317
Name: Engine Condition,
```

It indicate the class distribution of a binary target variable called "Engine Condition" in a DataFrame has the counts, the value "1" has a count of 12,317 instances, and the value "0" also has a count of 12,317 instances.
It represents that the dataset has been balanced, with an equal number of instances for both classes

Standardization of Data :

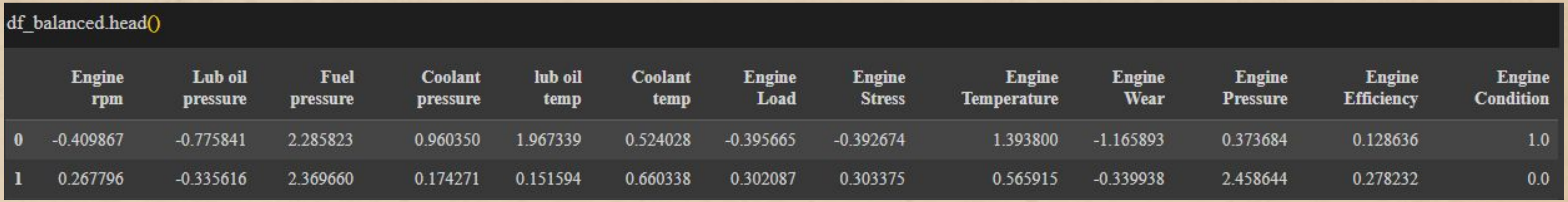


Standardization data is used to transform numerical data into a standardized scale. Data standardization bring all features onto the same scale, which can be beneficial for machine learning algorithms



```
std_scale=StandardScaler()
std_scale
```

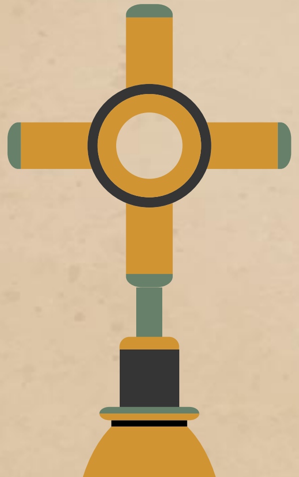
```
▾ StandardScaler
StandardScaler()
```



```
df_balanced.head()
```

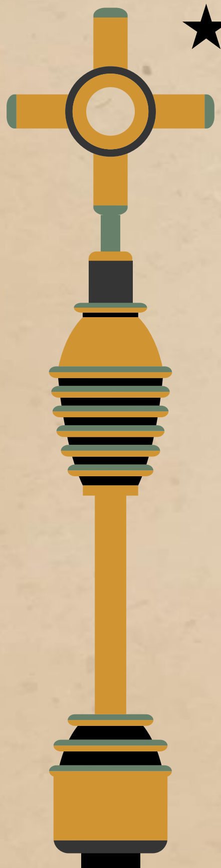
	Engine rpm	Lub oil pressure	Fuel pressure	Coolant pressure	lub oil temp	Coolant temp	Engine Load	Engine Stress	Engine Temperature	Engine Wear	Engine Pressure	Engine Efficiency	Engine Condition
0	-0.409867	-0.775841	2.285823	0.960350	1.967339	0.524028	-0.395665	-0.392674	1.393800	-1.165893	0.373684	0.128636	1.0
1	0.267796	-0.335616	2.369660	0.174271	0.151594	0.660338	0.302087	0.303375	0.565915	-0.339938	2.458644	0.278232	0.0

After standardizing the data, it becomes suitable for analysis as all features are transformed to the uniform scale.
This ensures that no individual feature dominates the learning process due to its larger scale, thereby preventing bias in the model.

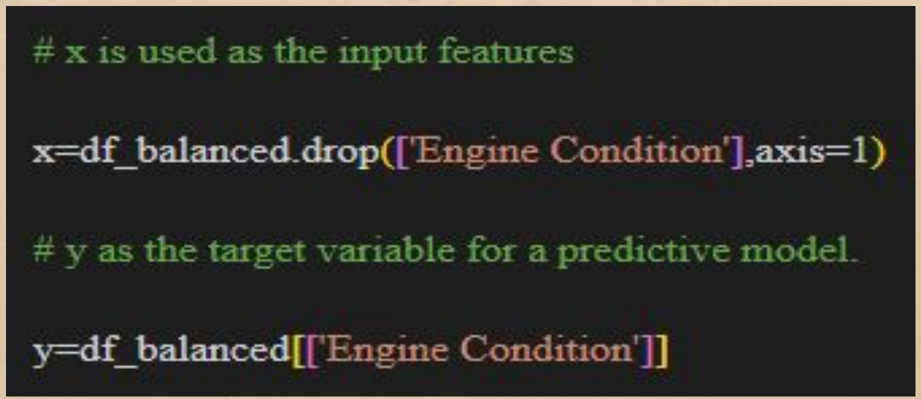


Model Building :

- ★ By building a machine learning model, historical data is utilized to train an algorithm that predicts the condition of an engine for unseen data.
- ★ Before model building splitting the Data Frame into input features and target variable
- ★ Splitting data into two separate subsets one for training the machine learning model and the other for evaluating its performance.

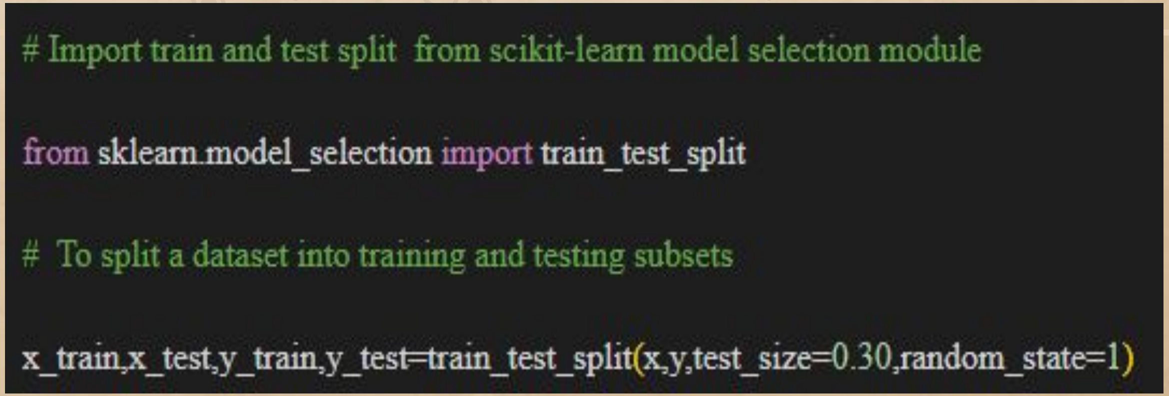


SPLITTING DATAFRAME INTO INPUT FEATURE & TARGET VARIABLE



```
# x is used as the input features

x=df_balanced.drop(['Engine Condition'],axis=1)

# y as the target variable for a predictive model.

y=df_balanced[['Engine Condition']]
```

SPLITTING INPUT FEATURE & TARGET VARIABLE INTO TRAIN DATA & TESTDATA



```
# Import train and test split  from scikit-learn model selection module

from sklearn.model_selection import train_test_split

#  To split a dataset into training and testing subsets

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=1)
```

Machine Learning Algorithm :

In this project, we aim to develop a predictive model for "Automobile Engine Health Prediction" by using various machine learning algorithms are,

☘️ Logistic Regression
☘️ Decision Tree Classifier
☘️ Random Forest Classifier
☘️ K-Nearest Neighbor Classifier
☘️ Support Vector Machine Classifier (SVMC)
☘️ XGBoost.

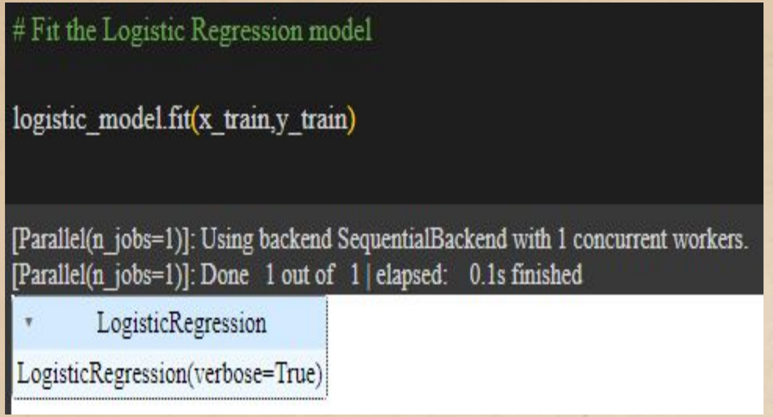
These Classification models are used when the goal is to predict the categorical or discrete class or category of a target variable based on input features.

We observed that the correlation between the target variable and independent variable is weak, indicating that the independent variables have minimal impact on the target variable. Therefore, we will proceed to build the model using two different approaches:

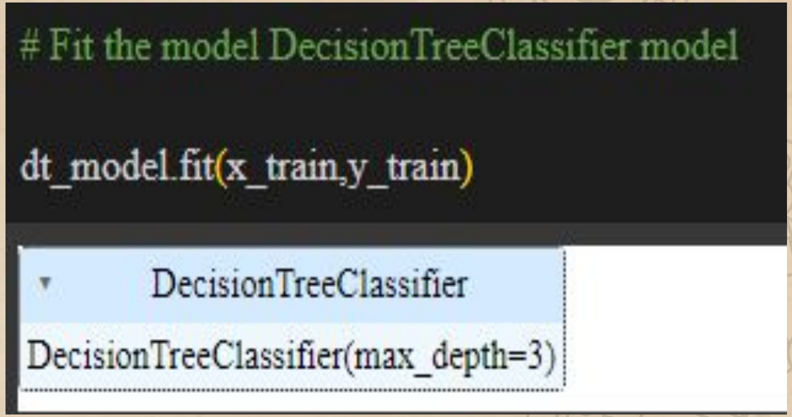
☘️ Approach 1 : Model with class balance and standardization of data
☘️ Approach 2 : Model with raw data using a pipeline (A pipeline will be created to streamline the modeling process).

To assess the data quality, we will examine the above-mentioned approaches.

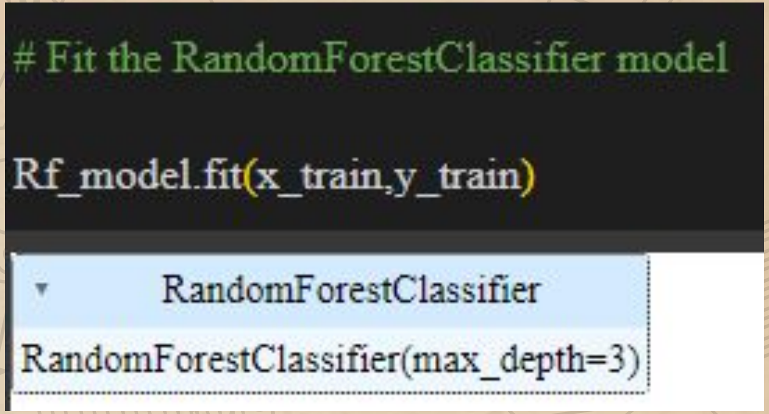
Approach 1 : Model with class balance and standardization of data



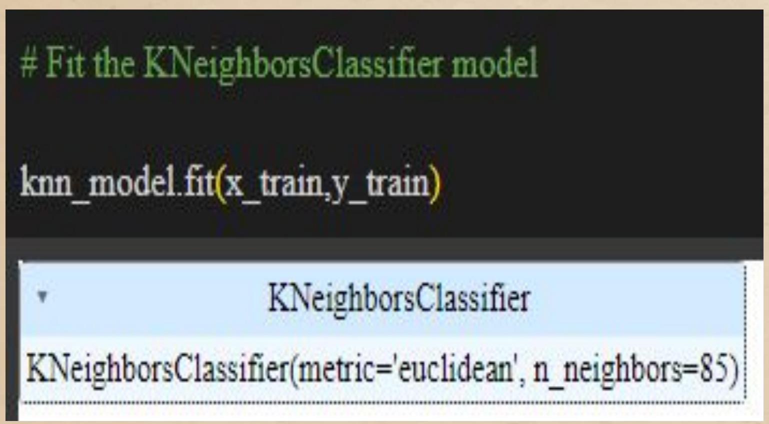
```
# Fit the Logistic Regression model

logistic_model.fit(x_train,y_train)

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s finished
▾ LogisticRegression
LogisticRegression(verbose=True)
```



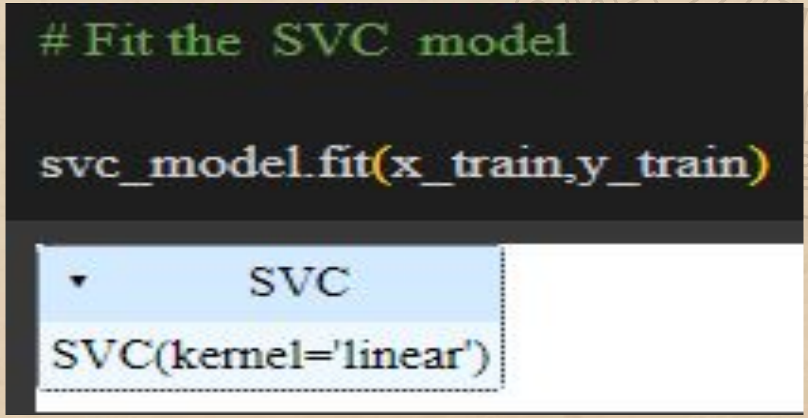
```
# Fit the model DecisionTreeClassifier model

dt_model.fit(x_train,y_train)

▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3)
```



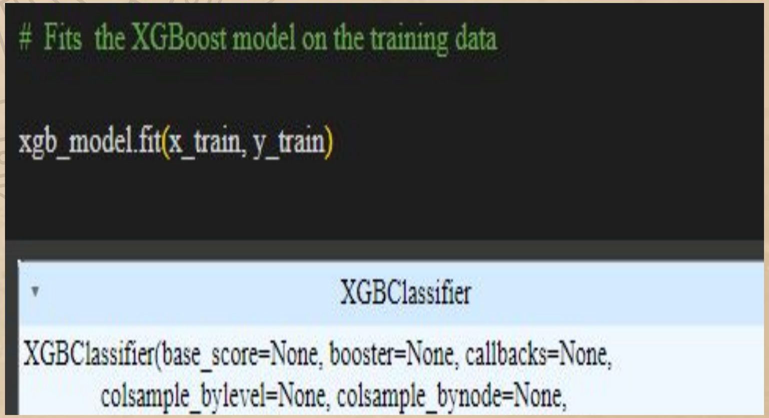
```
# Fit the RandomForestClassifier model

Rf_model.fit(x_train,y_train)

▾ RandomForestClassifier
RandomForestClassifier(max_depth=3)
```



```
# Fit the KNeighborsClassifier model

knn_model.fit(x_train,y_train)

▾ KNeighborsClassifier
KNeighborsClassifier(metric='euclidean', n_neighbors=85)
```



```
# Fit the  SVC  model

svc_model.fit(x_train,y_train)

▾ SVC
SVC(kernel='linear')
```

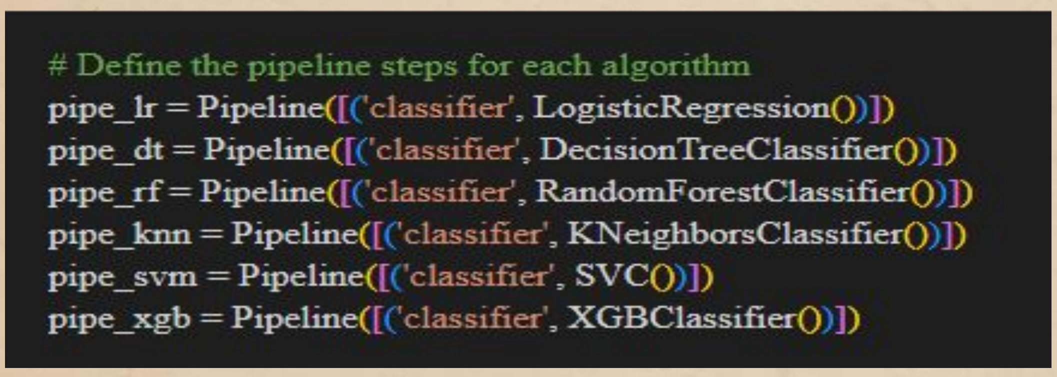


```
#  Fits  the XGBoost model on the training data

xgb_model.fit(x_train, y_train)

▾ XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
```

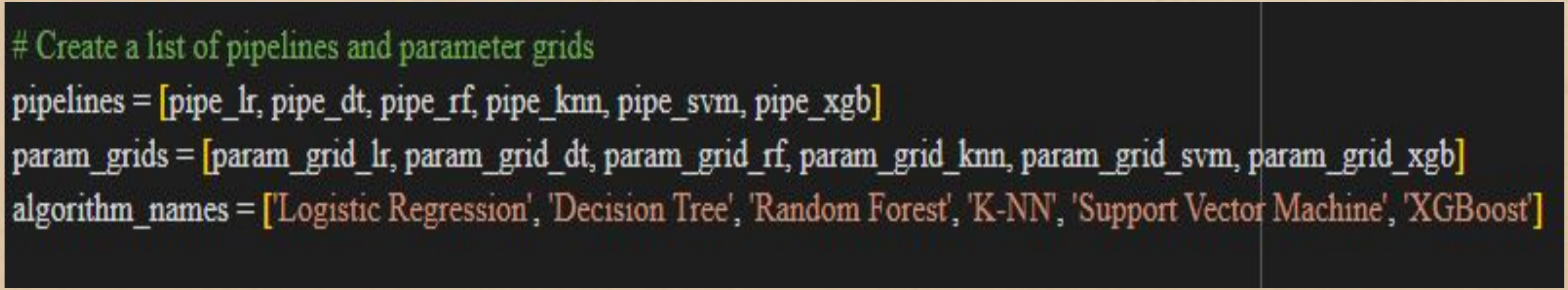
Approach 2 : Model with raw data using a pipeline

- ★ In machine learning, a pipeline refers to a series of data processing steps that are chained together in a specific order to facilitate the efficient and consistent execution of a machine learning workflow.

- ★ It combines multiple data transformation and modeling techniques into a single unit, allowing for streamlined and automated processes.



```
# Define the pipeline steps for each algorithm
pipe_lr = Pipeline([('classifier', LogisticRegression())])
pipe_dt = Pipeline([('classifier', DecisionTreeClassifier())])
pipe_rf = Pipeline([('classifier', RandomForestClassifier())])
pipe_knn = Pipeline([('classifier', KNeighborsClassifier())])
pipe_svm = Pipeline([('classifier', SVC())])
pipe_xgb = Pipeline([('classifier', XGBClassifier())])
```

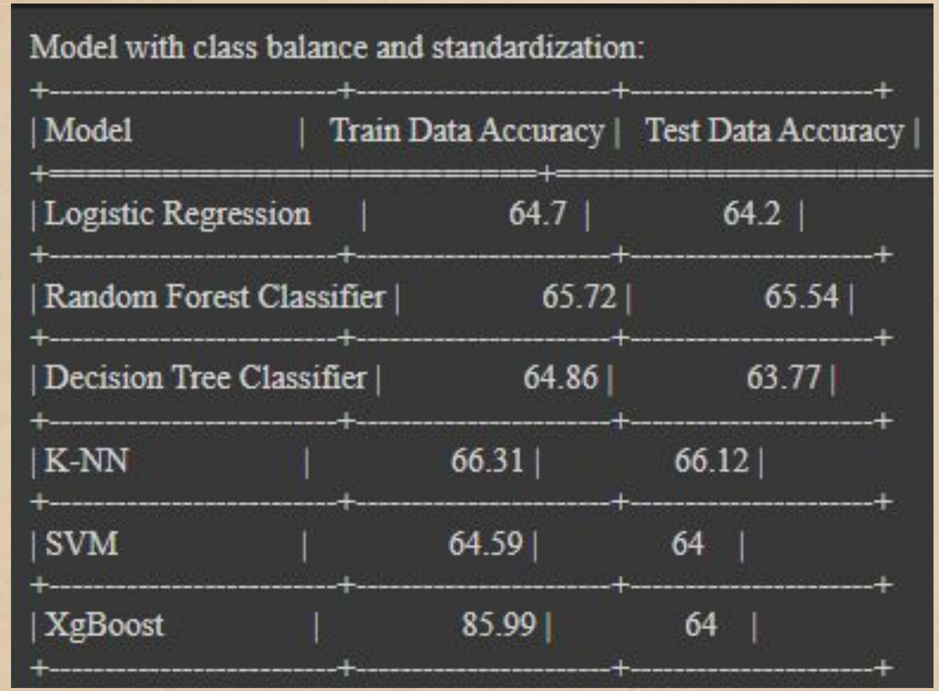


```
# Create a list of pipelines and parameter grids
pipelines = [pipe_lr, pipe_dt, pipe_rf, pipe_knn, pipe_svm, pipe_xgb]
param_grids = [param_grid_lr, param_grid_dt, param_grid_rf, param_grid_knn, param_grid_svm, param_grid_xgb]
algorithm_names = ['Logistic Regression', 'Decision Tree', 'Random Forest', 'K-NN', 'Support Vector Machine', 'XGBoost']
```

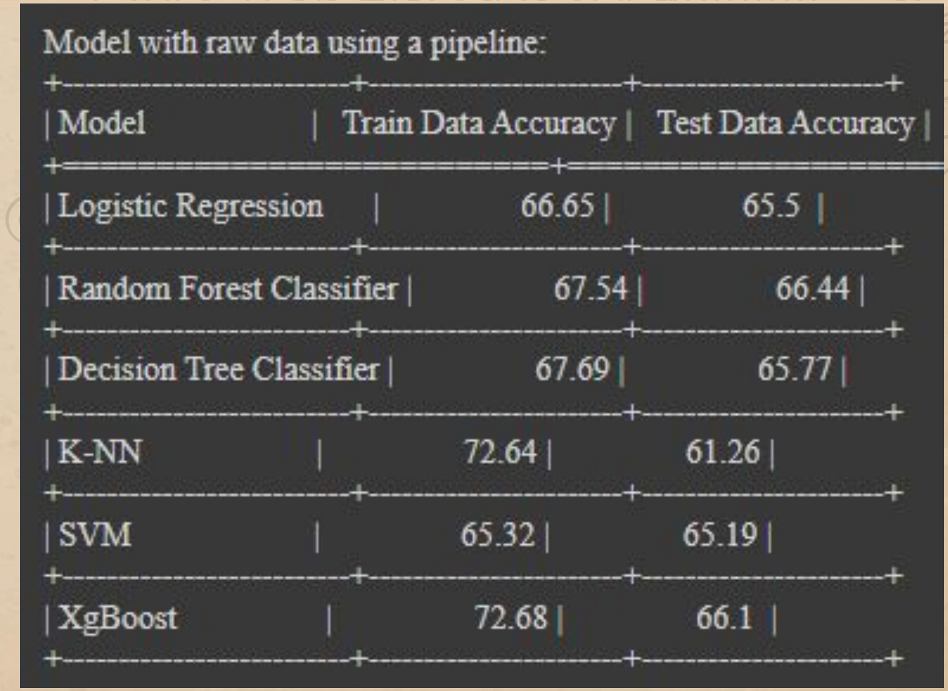
Accuracy of both approaches :

Approach 1:
Balanced & Standardised Data

Approach 2 :
Raw Data Using Pipeline



```
Model with class balance and standardization:
+--------------------------+-----------------------+-----------------------+
| Model                    | Train Data Accuracy | Test Data Accuracy |
+==========================+=======================+=======================
| Logistic Regression      |              64.7 |            64.2 |
+--------------------------+-----------------------+-----------------------+
| Random Forest Classifier |             65.72 |           65.54 |
+--------------------------+-----------------------+-----------------------+
| Decision Tree Classifier |             64.86 |           63.77 |
+--------------------------+-----------------------+-----------------------+
| K-NN                     |             66.31 |           66.12 |
+--------------------------+-----------------------+-----------------------+
| SVM                      |             64.59 |              64 |
+--------------------------+-----------------------+-----------------------+
| XgBoost                  |             85.99 |              64 |
+--------------------------+-----------------------+-----------------------+
```



```
Model with raw data using a pipeline:
+--------------------------+-----------------------+-----------------------+
| Model                    | Train Data Accuracy | Test Data Accuracy |
+==========================+=======================+=======================
| Logistic Regression      |             66.65 |            65.5 |
+--------------------------+-----------------------+-----------------------+
| Random Forest Classifier |             67.54 |           66.44 |
+--------------------------+-----------------------+-----------------------+
| Decision Tree Classifier |             67.69 |           65.77 |
+--------------------------+-----------------------+-----------------------+
| K-NN                     |             72.64 |           61.26 |
+--------------------------+-----------------------+-----------------------+
| SVM                      |             65.32 |           65.19 |
+--------------------------+-----------------------+-----------------------+
| XgBoost                  |             72.68 |            66.1 |
+--------------------------+-----------------------+-----------------------+
```

Conclusion :

★ The results obtained from our machine learning models indicate,

☘️ Logistic Regression, Random Forest Classifier, and Decision Tree Classifier demonstrate better performance when using raw data with a pipeline, as opposed to class balance and standardized data.

☘️ K-NN performs better with class balance and standardized data.

☘️ SVM demonstrates comparable performance in terms of accuracy between the two approaches. However, the model with raw data using a pipeline shows improved accuracy compared to the model with standardized balanced data.

☘️ XGBoost achieves high accuracy on the train data with class balance and standardized data but performs better overall with raw data using a pipeline.

★ Overall, the accuracy of the models is good, but there is still room for improvement. One way to improve the accuracy of the models would be to collect more data with more different set of features may be able to make better predictions.

★ In terms of a business perspective, the accuracy of the models is important because it determines how well the models can be used to make decisions. If the models are not accurate, then they may not be able to make good decisions. This could lead to lost profits or other negative consequences.

Thank you