ENGINE HEALTH ANALYTICS & MODELING WITH DATASCIENCE &MACHINE LEARNING

Ruchi Joshi

List of Figures

List of Tables

List of Equations:

Understanding Business Problem

Defining the Problem Statement

Context:

Vehicle breakdowns and engine failures lead to significant financial losses for both individual owners and fleet operators. Unexpected engine failures can cause expensive repairs, operational downtime, and safety risks. Predictive maintenance in the automotive industry can help minimize these issues by leveraging sensor data to forecast potential failures before they occur.

Automobile manufacturers, fleet managers, and service providers aim to develop data-driven solutions to improve engine reliability and optimize maintenance schedules. By analyzing engine health parameters such as RPM, temperature, pressure, and other sensor readings, machine learning models can be trained to predict when an engine requires maintenance, allowing proactive intervention before a failure occurs.

Objective:

As a Data Scientist, our goal is to build a predictive maintenance model that can analyze historical and real-time engine sensor data to identify potential failures. The model should accurately classify whether an engine requires maintenance or is operating normally.

This solution will help:

- Reduce unplanned breakdowns and costly repairs.
- Improve vehicle performance and engine lifespan.
- Optimize maintenance schedules to minimize downtime.
- Provide data-driven insights to manufacturers and fleet operators for better decision-making.

Need of the Study/Project

Unplanned engine failures are a major challenge for both individual vehicle owners and fleet operators, leading to severe financial and operational losses. On average, a single day of vehicle downtime can result in losses ranging from ₹1–2 lakh per truck per day, considering missed delivery schedules, contractual penalties, and idle manpower.

Moreover, frequent breakdowns increase maintenance costs by up to 20–25% annually and significantly reduce engine lifespan due to cumulative wear and thermal stress. In large-scale fleet operations, unexpected engine failures can lead to loss of customer trust, productivity disruptions, and safety incidents.

Traditional preventive maintenance — where servicing is done at fixed intervals — often fails to capture real-time wear patterns and may lead to either over-servicing (unnecessary costs) or under-servicing (unexpected failures).

Hence, there is a strong need for a data-driven predictive maintenance approach, leveraging engine sensor data (Engine RPM, Lube oil pressure, fuel pressure, Coolant pressure, Lub Oil temperature, Coolant temperature, etc.) to forecast potential failures before they occur. Studies and industry pilots suggest that predictive maintenance can reduce unplanned breakdowns by 30–40%, lower maintenance costs by up to 25%, and extend engine lifespan by 20% or more.

By building a machine learning model that analyzes real-time sensor data, the automotive sector can shift from reactive to proactive maintenance as shown in the flow chart below — ensuring operational continuity and optimal resource utilization.

Sensor Data (RPM, Pressure, Temperature Variables)

↓

Data Processing C Model Training

↓

Predictive Maintenance Model (Machine Learning)

↓

Early Warning Alerts to Operators

↓

Scheduled Maintenance C Cost Savings

Understanding business/social opportunity

Implementing predictive maintenance in the automotive industry offers a multi-dimensional opportunity spanning business, safety, and sustainability outcomes:

- Operational Efficiency - Reduction in unplanned breakdowns leads to lower downtime costs and improved fleet utilization.
- Financial Impact - Optimized maintenance schedules can save ₹5–10 lakh annually per fleet of 50 vehicles through reduced part replacements and labor costs.
- Safety Improvement - Early fault detection minimizes the risk of on-road breakdowns and accidents caused by engine overheating or mechanical failure.
- Environmental Sustainability - Fewer component replacements and optimized fuel efficiency contribute to reduced carbon footprint and material waste.
- Data-Driven Decision Making - Enables manufacturers (OEMs) and service providers to derive actionable insights from sensor data, improving design, warranty analytics, and service planning.

EDA

Data collection and background: Data was provided by Great Learning

Data Overview

The sensor values in the dataset are consistent with the operating parameters of larger and small engines commonly found in equipment like vehicles, lawnmowers, portable generators, and compact machinery. Some engines operate at lower RPMs, pressures, and temperatures compared to larger automotive engines, and vice versa. Therefore, the data is appropriate for developing predictive maintenance models tailored to large and small engine applications.

The detailed data dictionary is given below:

Data Description:

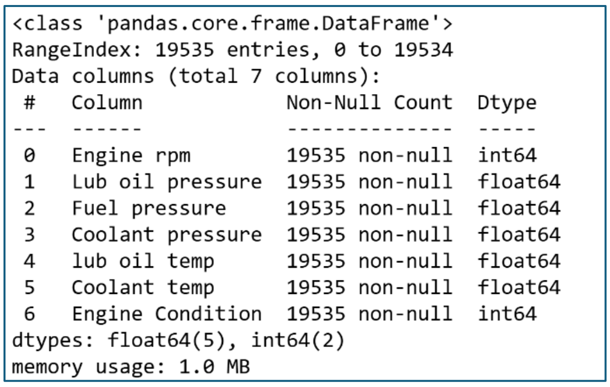
- Data has 19535 rows C 7 columns with Column names (**Data Dictionary**) as under:
- **Engine_RPM**: The number of revolutions per minute (RPM) of the engine, indicating engine speed. It is defined in Revolutions per Minute (RPM).
- **Lub_Oil_Pressure**: The pressure of the lubricating oil in the engine, essential for reducing friction and wear. It is defined in bar or kilopascals (kPa)
- **Fuel_Pressure**: The pressure at which fuel is supplied to the engine, critical for proper combustion. It is defined in bar or kilopascals (kPa)
- **Coolant_Pressure**: The pressure of the engine coolant, affecting engine temperature regulation. It is defined in bar or kilopascals (kPa)
- **Lub_Oil_Temperature**: The temperature of the lubricating oil, which impacts viscosity and engine performance. It is defined in degrees Celsius (°C)
- **Coolant_Temperature**: The temperature of the engine coolant, crucial for preventing overheating. It is defined in degrees Celsius (°C)
- **Engine_Condition**: A categorical or numerical label representing the health of the engine, potentially indicating normal operation or various levels of wear and failure risks. It is defined as a categorical variable (0/1) representing a state such as "0 = Off/False/Active" and "1 = On/True/Faulty"

Understanding Data Types, Data Irregularities, Missing Values s Treatment:

- **Data had no missing values**

- Engine rpm is integer (Fig 1)
- Engine Condition is target variable 0/1 type for Model Building

- All others are float
- **No Duplicates were found**



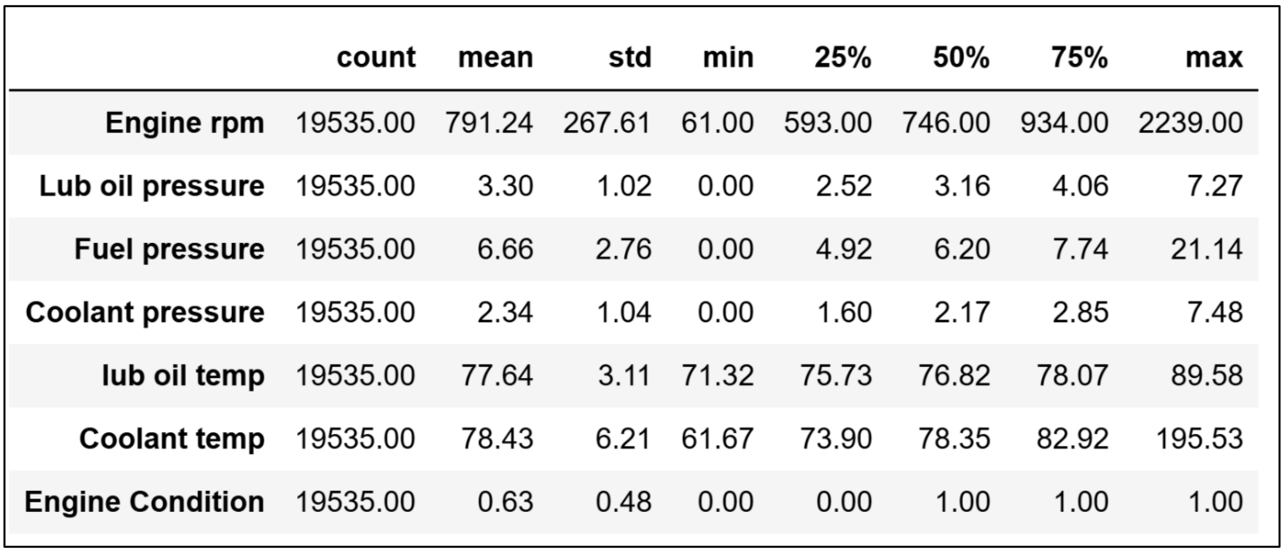
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19535 entries, 0 to 19534
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Engine rpm        19535 non-null  int64  
 1   Lub oil pressure  19535 non-null  float64
 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
```

Fig 1: Data Information

Statistical Summary:

Observations and Insights:

Refer Fig 2 below.



	count	mean	std	min	25%	50%	75%	max
Engine rpm	19535.00	791.24	267.61	61.00	593.00	746.00	934.00	2239.00
Lub oil pressure	19535.00	3.30	1.02	0.00	2.52	3.16	4.06	7.27
Fuel pressure	19535.00	6.66	2.76	0.00	4.92	6.20	7.74	21.14
Coolant pressure	19535.00	2.34	1.04	0.00	1.60	2.17	2.85	7.48
lub oil temp	19535.00	77.64	3.11	71.32	75.73	76.82	78.07	89.58
Coolant temp	19535.00	78.43	6.21	61.67	73.90	78.35	82.92	195.53
Engine Condition	19535.00	0.63	0.48	0.00	0.00	1.00	1.00	1.00

Fig 2. '5-point Statistics' of Numerical Data

- Engine rpm varies from 61 to 2239 with mean at 791 and median at 746, thus values are right skewed

- Lub oil pressure varies from 0.003 to 7.266 with mean at 3.3 and median at 3.1, thus values are slightly right skewed
- Fuel pressure varies from 0.003 to 21.1 with mean at 6.7 and median at 6.2, thus values are slightly right skewed
- Coolant pressure varies from 0.002 to 7.5 with mean at 2.3 and median at 2.2, thus values are slightly right skewed
- lub oil temp varies from 71.3 to 89.6 with mean at 77.6 and median at 76.8, thus values are slightly right skewed
- Coolant temp varies from 61.7 to 195.5 with mean at 78.4 and median at 78.3, thus values are slightly right skewed
- Engine Condition is '1' or faulty 63% times and '0' or active remaining times

(Please Refer Jupyter notebook (.ipynb file) for detailed value counts data of discrete integer numeric variables if needed)

Univariate Analysis

1. Engine rpm:

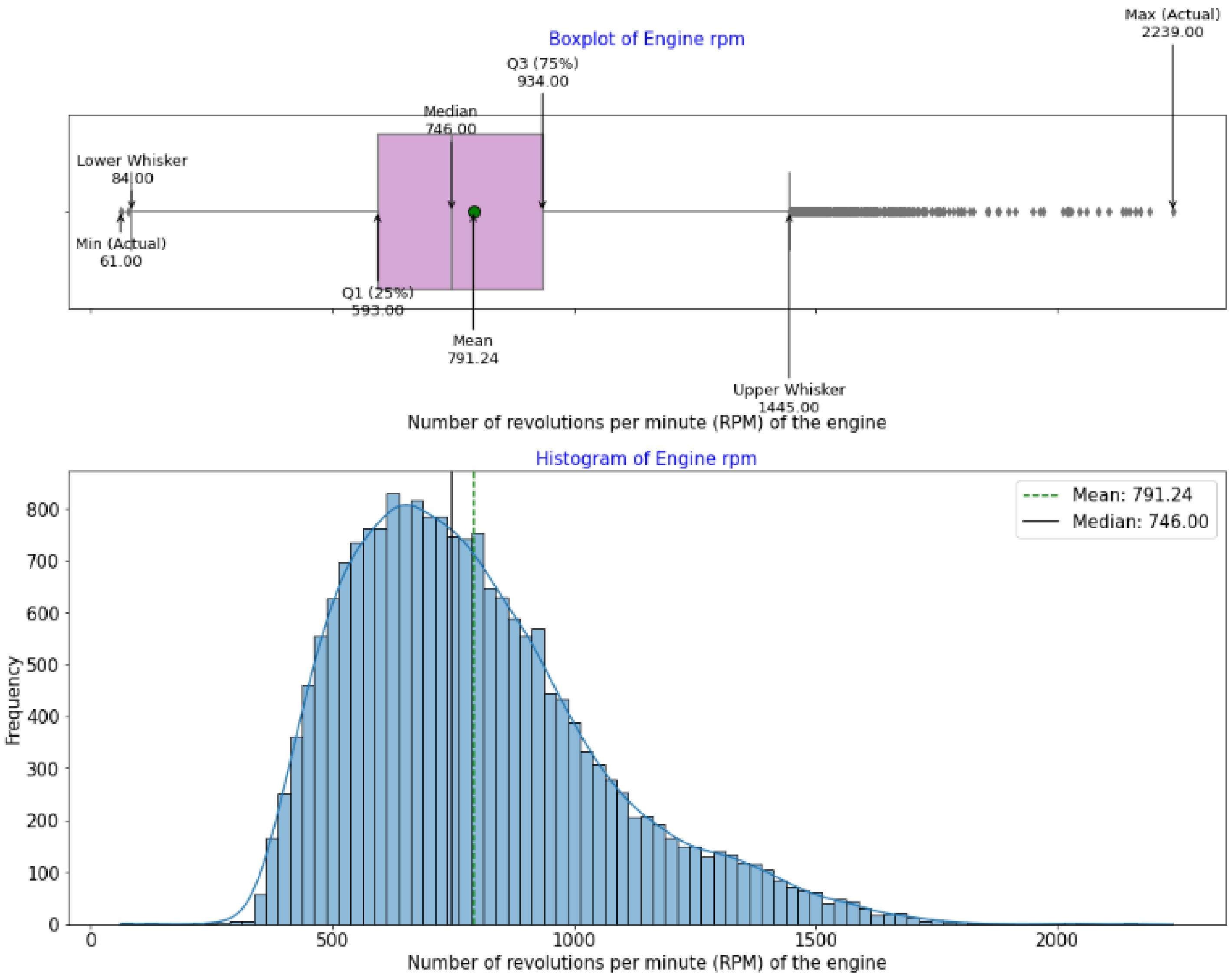
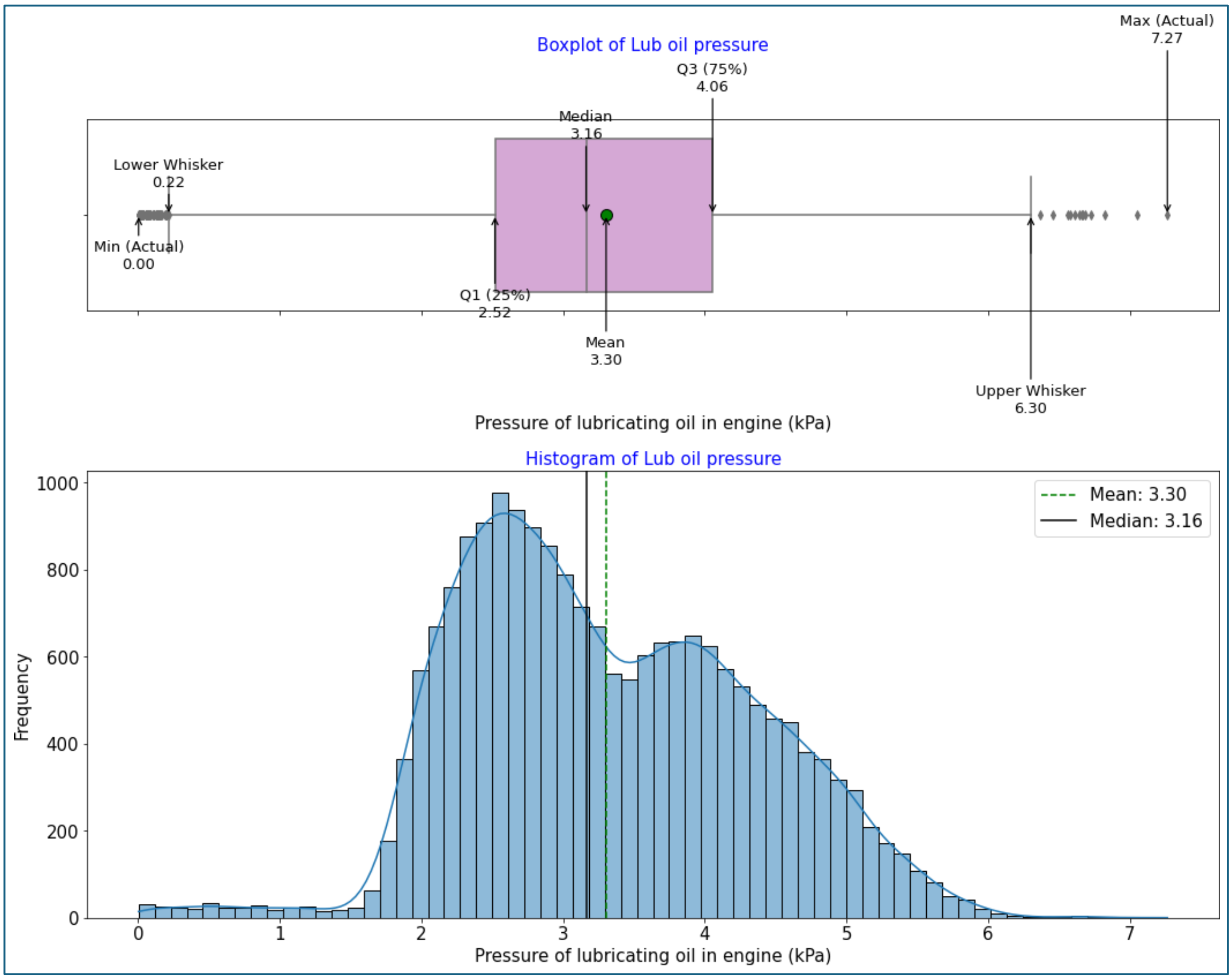


Fig 3: Histogram Boxplot of Engine rpm

- Data is widespread from 61 Rpm to 2239 Rpm (Fig 3)
- Data is right skewed with several outliers, especially at the higher end, which we wont treat as they are continuous real values
- 75% data lies below 934 Rpm

- Mean at 791 Rpm and median at 746 Rpm

2. Lub oil pressure:



- Distribution is bimodal with Mean at 3.3 KPa and median at 3.2 KPa and varies from 0 KPa to 7.27 KPa
- Data is slightly right skewed (Fig 4) with outliers at both ends (We won't be treating them as these are real values)

Fig 4: Boxplot s Histogram of Lub oil pressure

3. Fuel pressure :

- Data varies from 0 KPa to 21.14 KPa, with Mean at 6.7 KPa and median at 6.2 Kpa (Fig 5)
- Data is right skewed with outliers at both ends (We won't be treating them as these are real values)

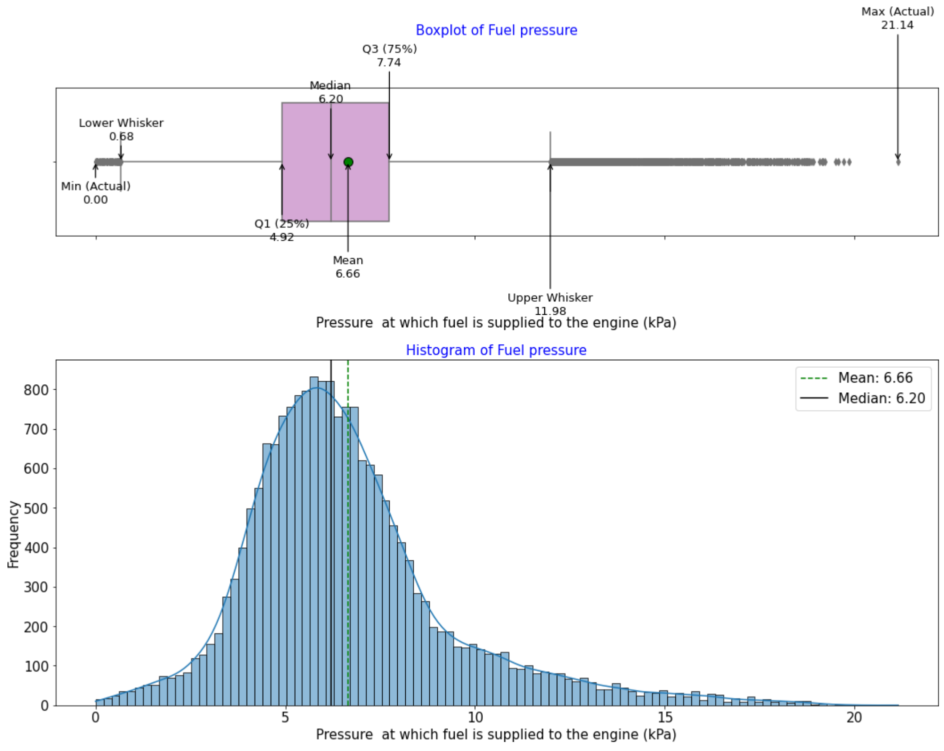


Fig 5: Boxplot s Histogram of Fuel pressure

4. Coolant pressure :

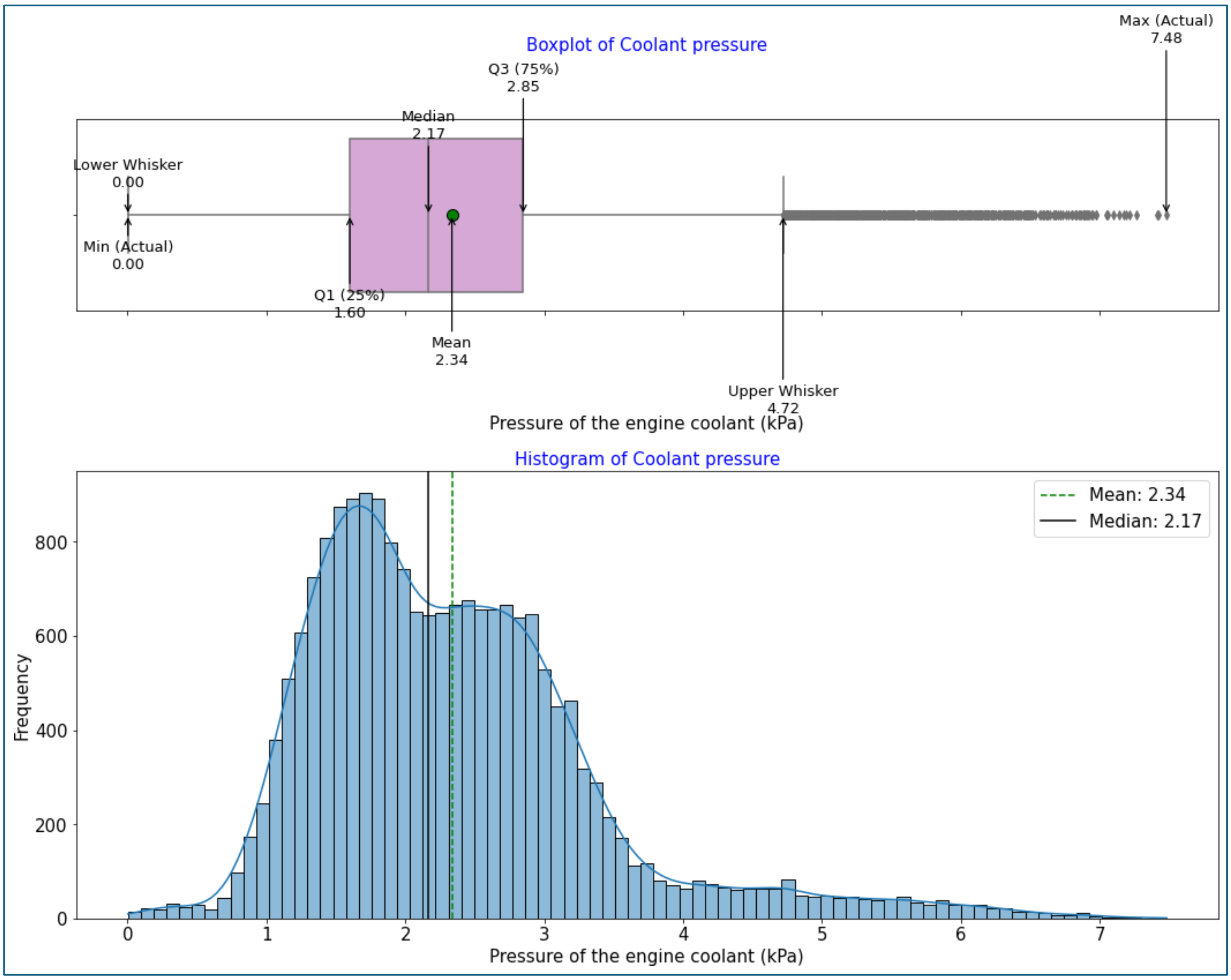


Fig 6: Box plot s Histogram of Coolant pressure

- Bimodal distribution with mean at 2.34 KPa and median at 2.17 Kpa (Fig 6)
- Data varies from 0 KPa to 7.48 KPa
- Data is right skewed with several outliers, especially on the higher end (We won't be treating them as these are real values)
- 75% data is below 2.85 KPa

5. lub oil temp:

- Data is highly right skewed with alot of outliers, especially on higher end (We won't be treating them as these are real values)
- Varies from 71°C to 89°C with mean at 78°C and median at 77°C (Fig 7)

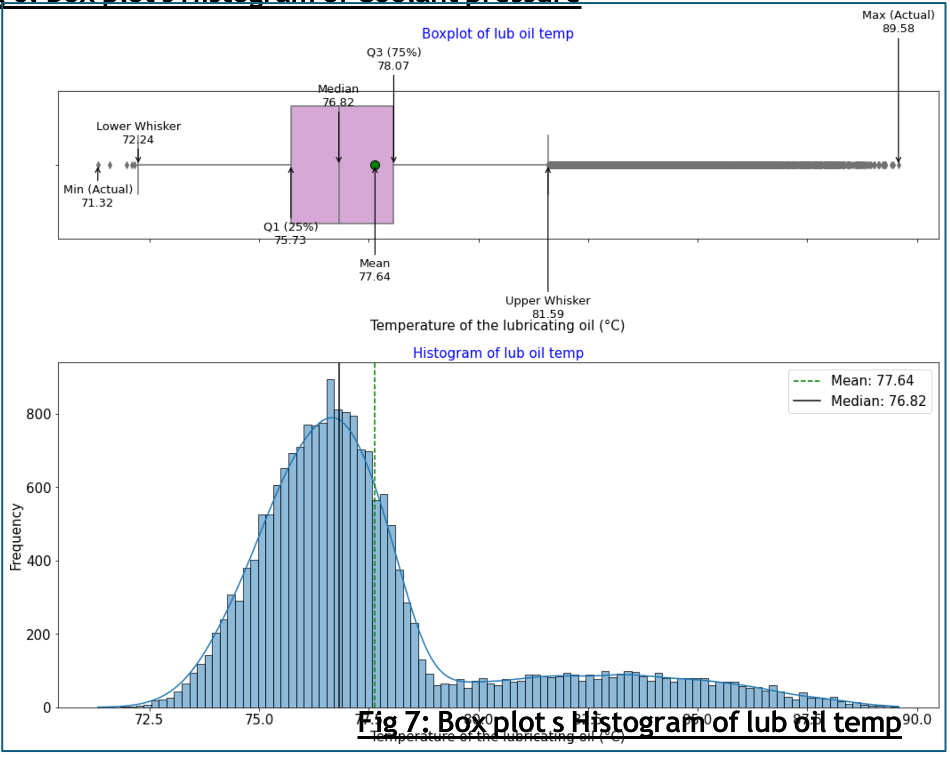
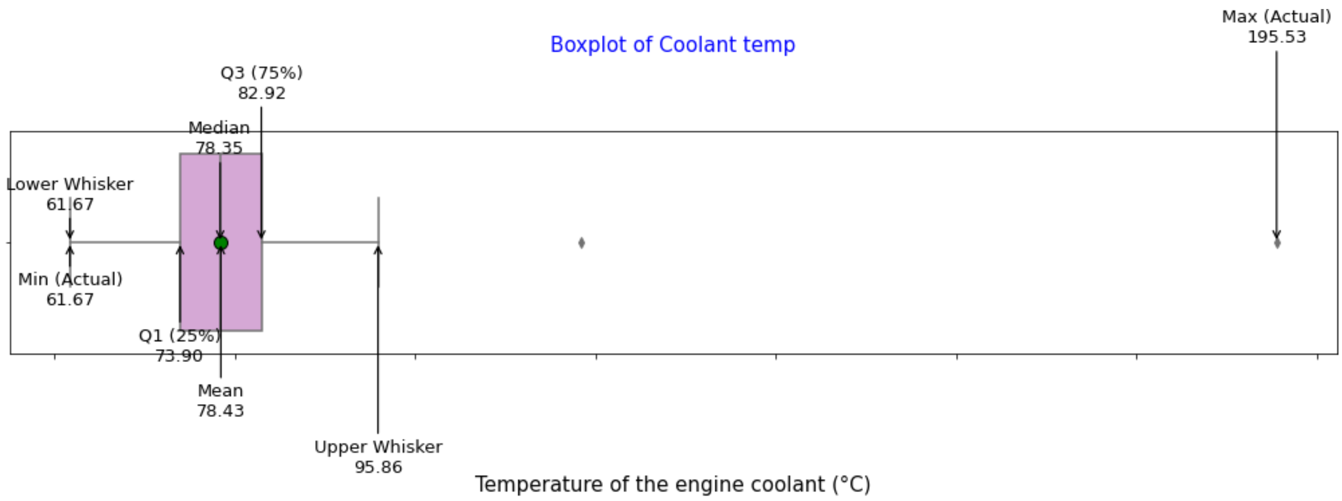


Fig 7: Box plot s Histogram of lub oil temp

- 75% data lies below 78°C

6. Coolant temp :



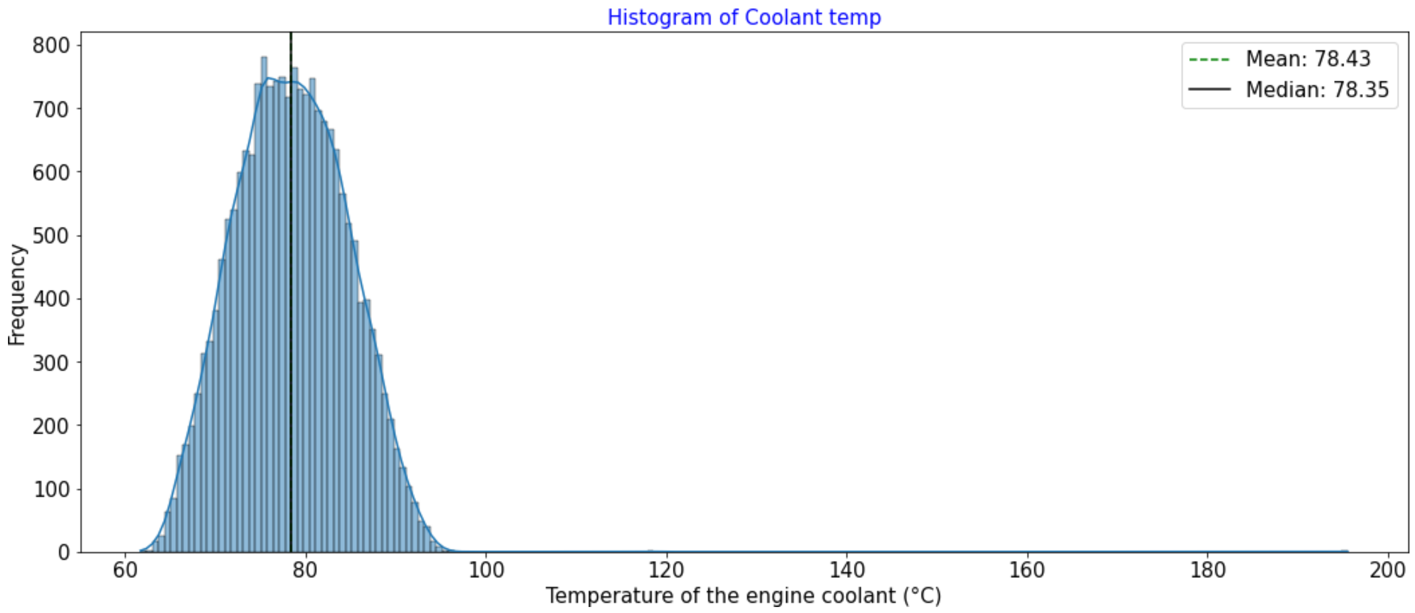


Fig 8: Box plot s Histogram of Coolant temp

	Engine rpm	Lub oil pressure	Fuel pressure	Coolant pressure	lub oil temp	Coolant temp	Engine Condition
2423	455	2.01	7.81	1.62	76.52	195.53	1
15123	760	4.91	2.59	1.75	77.87	118.37	0

Fig G : Outliers of Coolant temp

- Data varies from 61°C to 195 °C with mean and median at 78°C (Fig 8)
- Data is normally distributed C has just 2 outliers (Fig 9) , but We won't be treating them as these are real values
- 75% data lies below 83 °C

7. Engine Condition:

63.1% Engines are faulty (Fig 10)

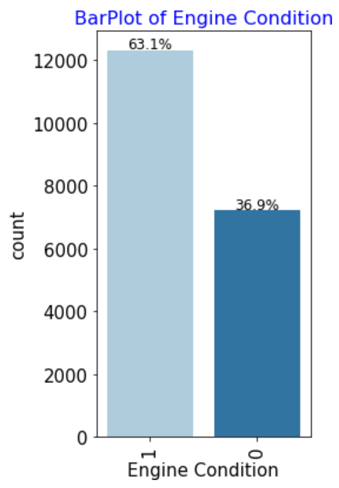


Fig 10 : Barplot of Engine Condition

Bivariate Analysis

Understanding correlation between all numerical variables:

As seen in Pairplot (Fig 12 below) and Heatmap (Fig 11 besides)

- Engine Condition is the target variable so we will ignore it for now and conclude based on our Baseline Model later
- There is no correlation among the numerical variables, which is good for the model
- No other pattern is seen in the Independent variables , which is good for our analysis

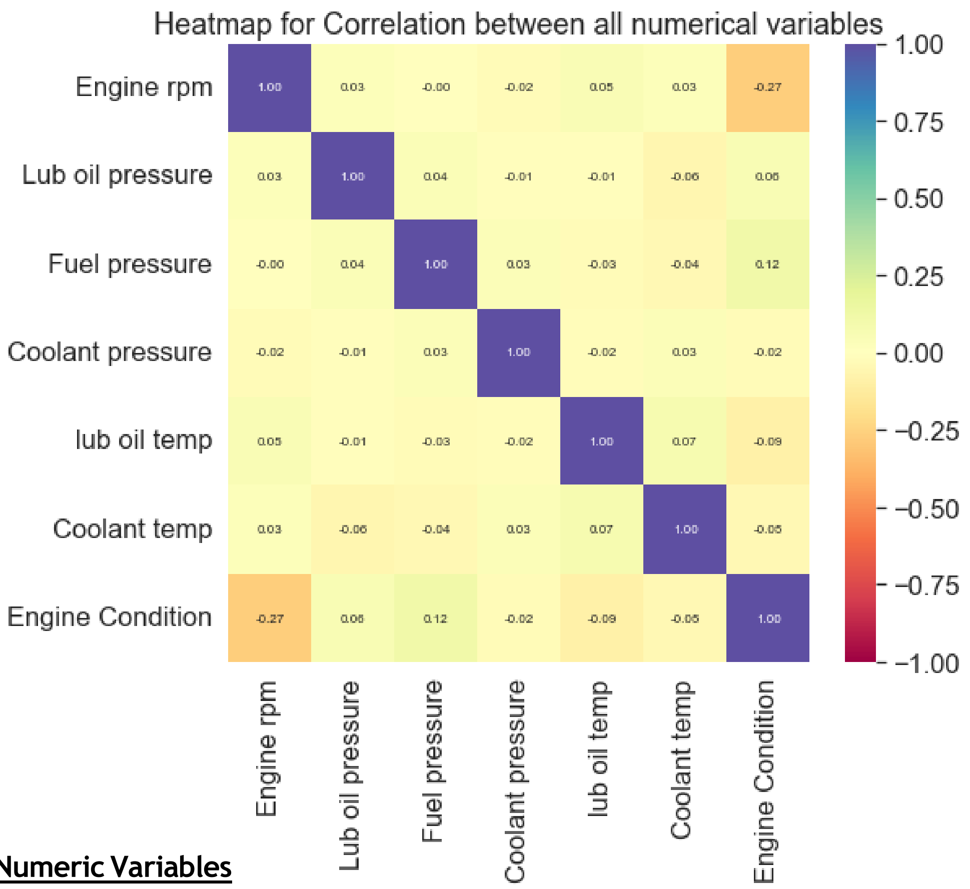


Fig 11: Correlation Plot of all Numeric Variables

Explore the relationship between all numerical variables:

Understanding Predictor vs Target Variables:

1. Engine rpm wrt Engine Condition:

There is a clear difference in Engine rpm values for Active and faulty engines with Engine Rpm being considerably less for faulty engines (Refer Fig 13 below)

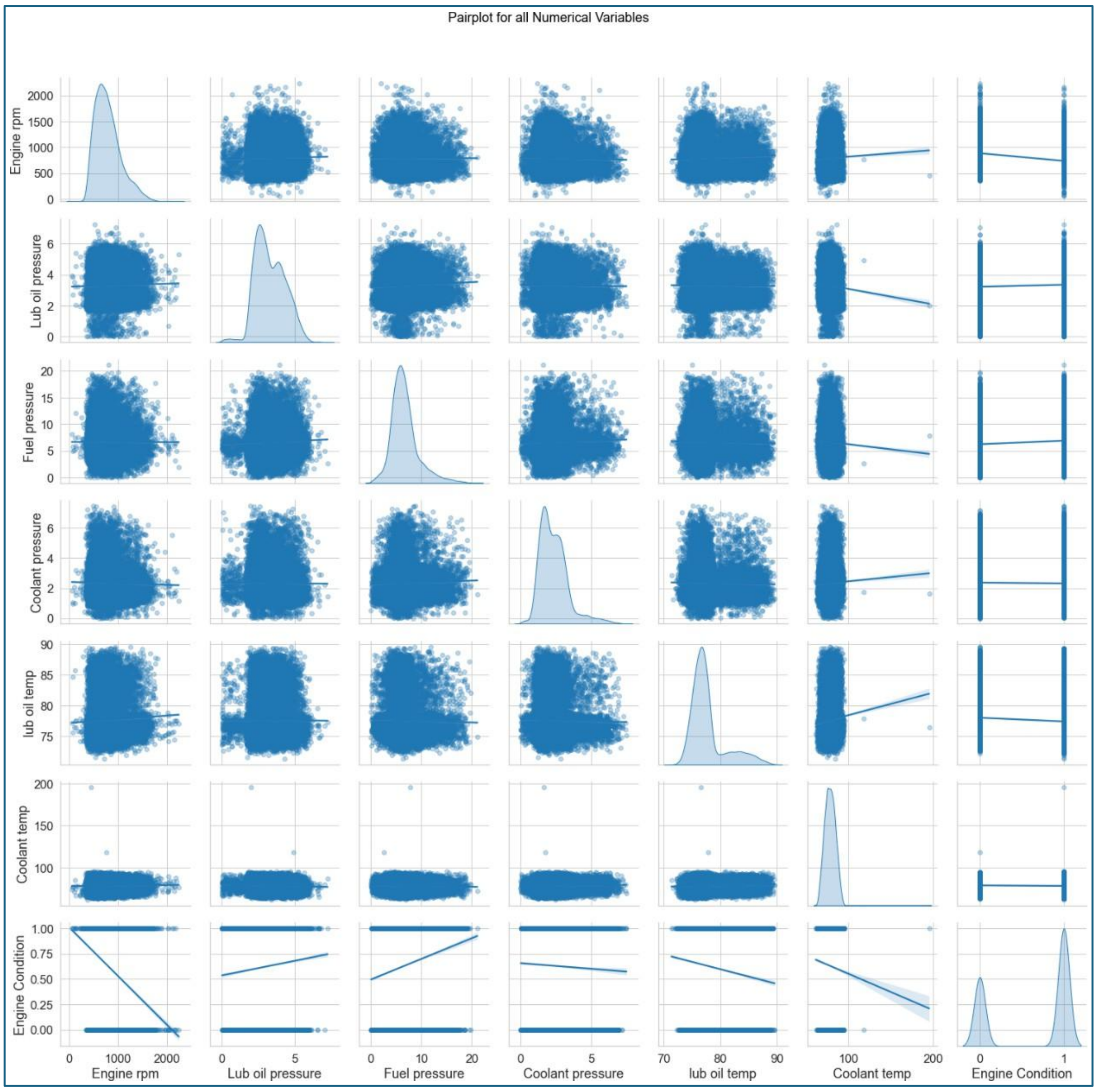


Fig 12: Pair Plot of all Numeric Variables

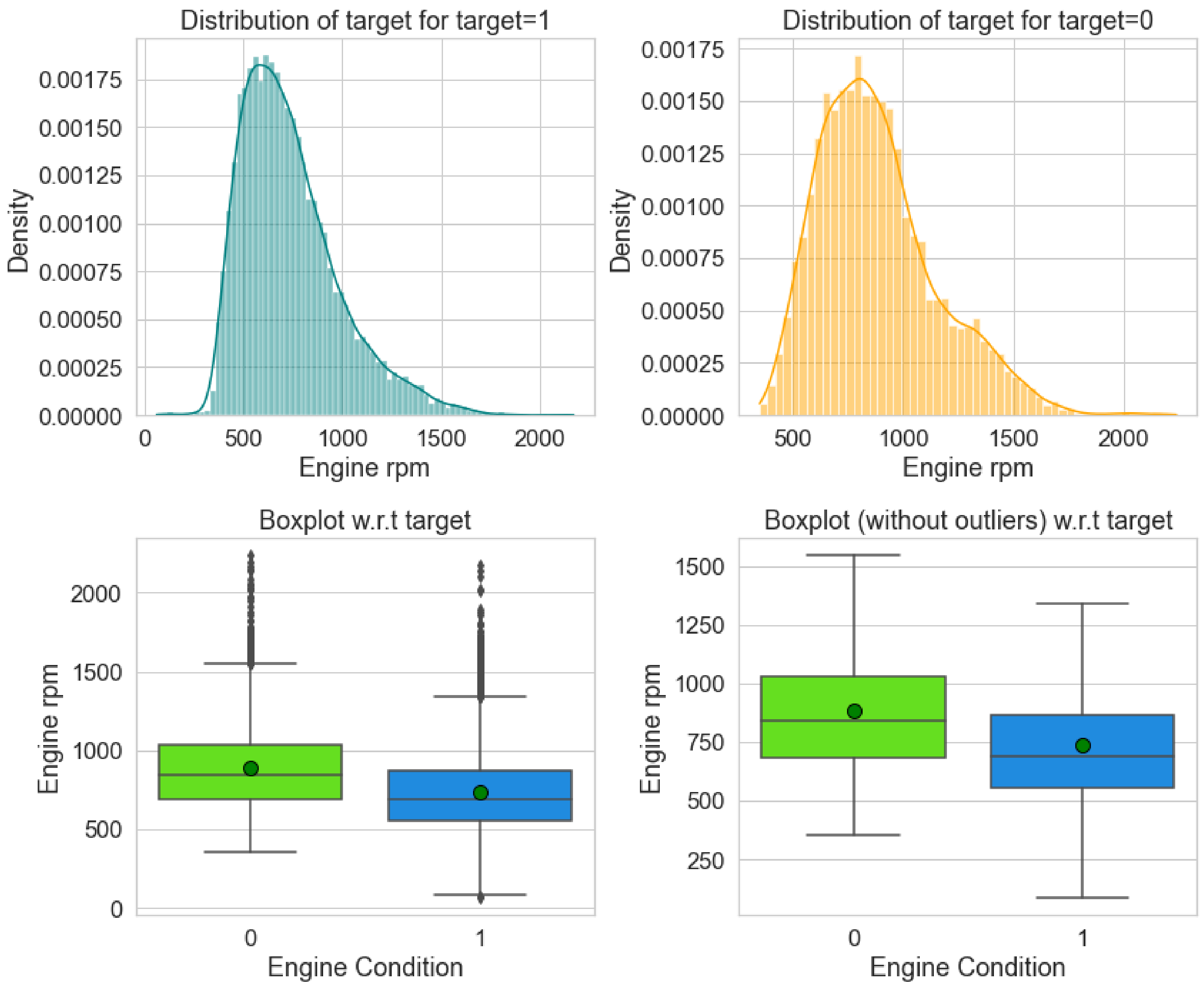


Fig 13: Distribution of Engine rpm wrt Engine Condition

2. *Lub oil pressure wrt Engine Condition*:

- Lube oil pressure (KPa) is slightly higher for Faulty Engines, though the difference is very less (Fig 14)

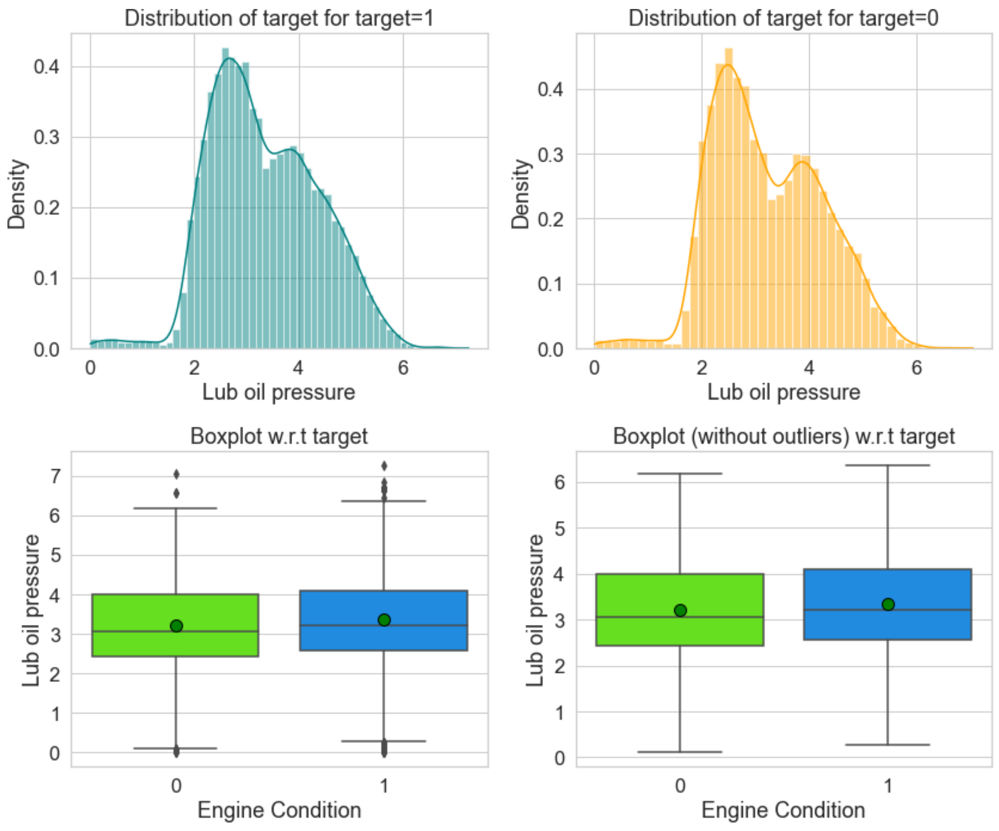
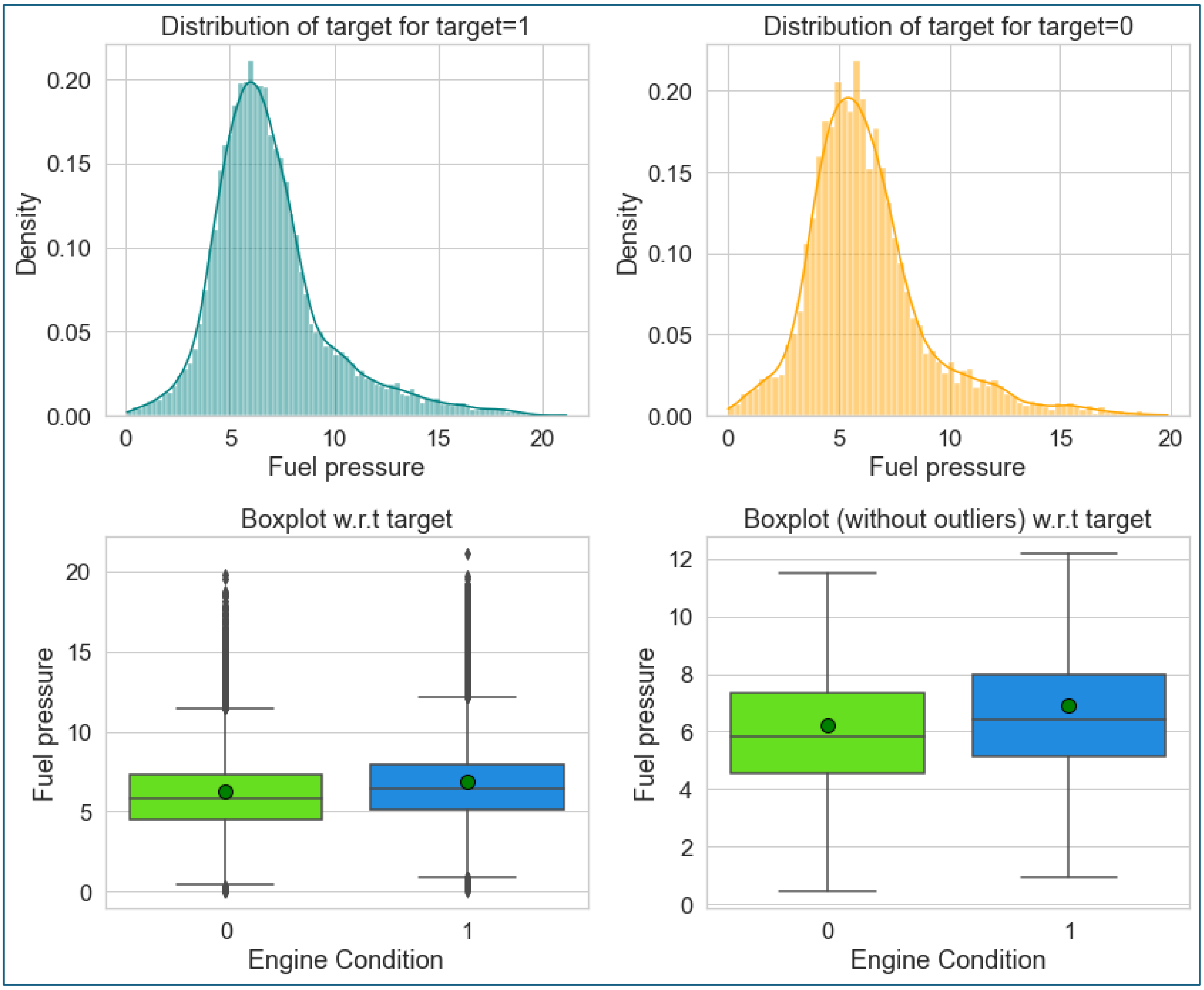
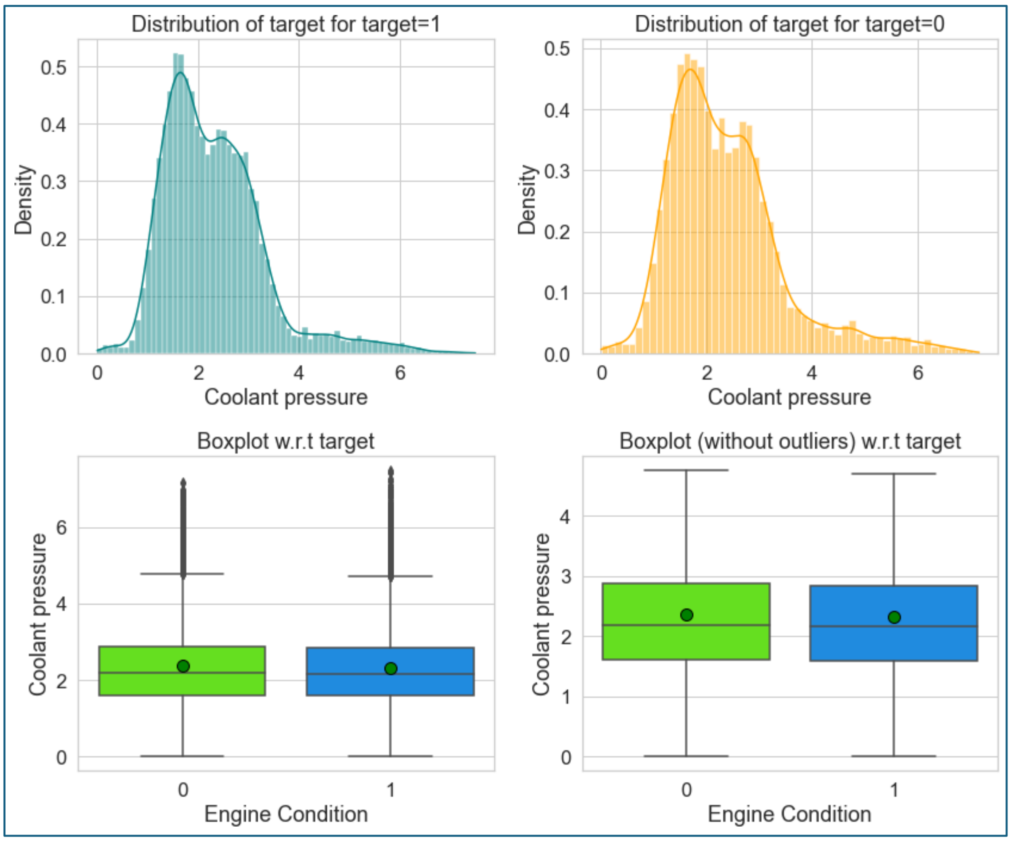


Fig 14: Distribution of Lub oil pressure wrt Engine Condition

3. Fuel pressure wrt Engine Condition:





<u>Fig 15: Distribution of Fuel pressure wrt Engine Condition</u>

Fuel pressure (KPa) is slightly higher for Faulty Engines, though the difference is less (Fig 15)

4. Coolant pressure wrt Engine Condition:

There is no visible difference between coolant pressures of Active C Faulty Engines (Fig 16)

<u>Fig 16: Distribution of Coolant pressure wrt Engine</u>

5. lub oil temp wrt Engine Condition:

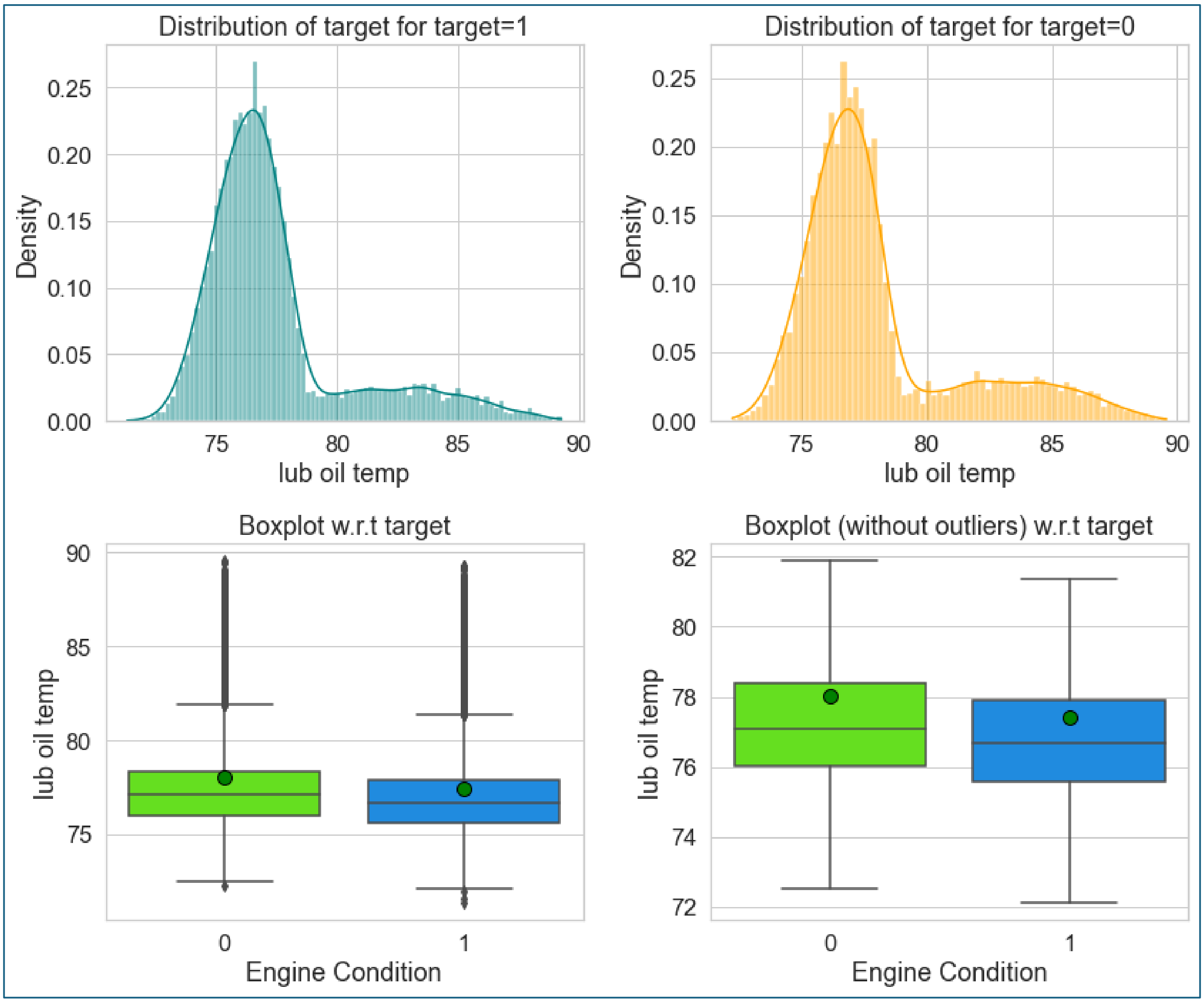


Fig 17: Distribution of lub oil temp wrt Engine Condition

Lube oil temp is slightly less for faulty engines, though the difference is less (Fig 17)

c. Coolant temp wrt Engine Condition:

If we don't consider the 2 outliers, Coolant temp is slightly less for Faulty engines , though the difference is very less (Fig 18)

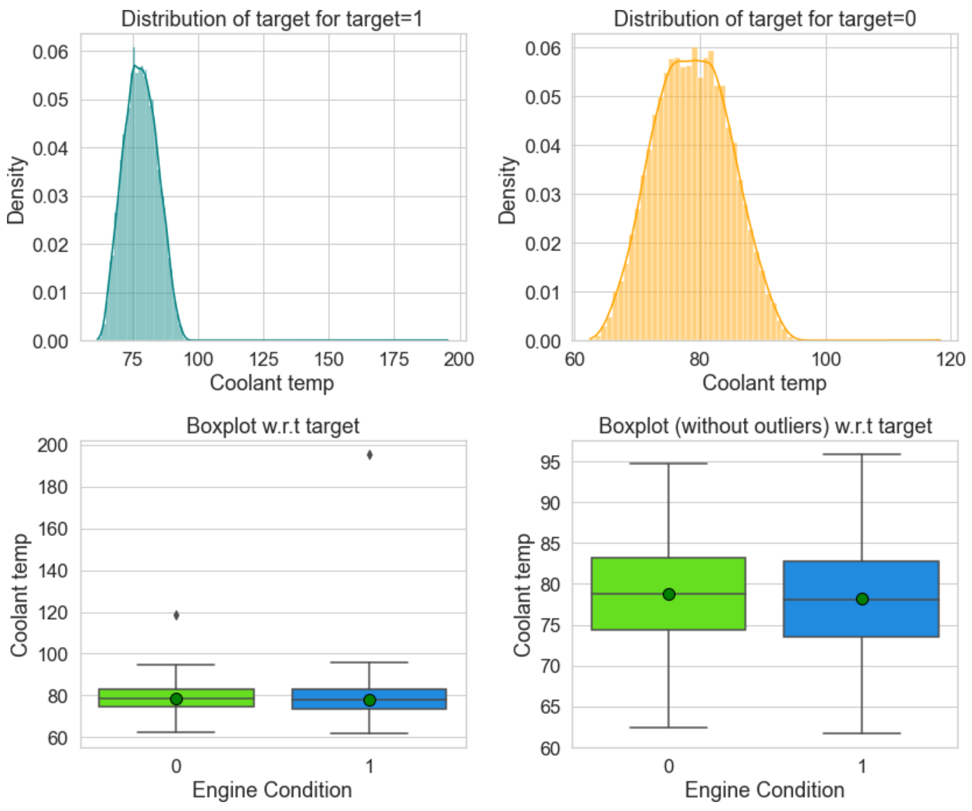


Fig 18: Distribution of Coolant temp wrt Engine Condition

Multivariate Analysis Using PCA:

To better understand the underlying structure of the dataset and reduce redundancy among correlated variables, Principal Component Analysis (PCA) was applied. PCA is a statistical technique used to reduce the dimensionality of data while preserving as much variance (information) as possible.

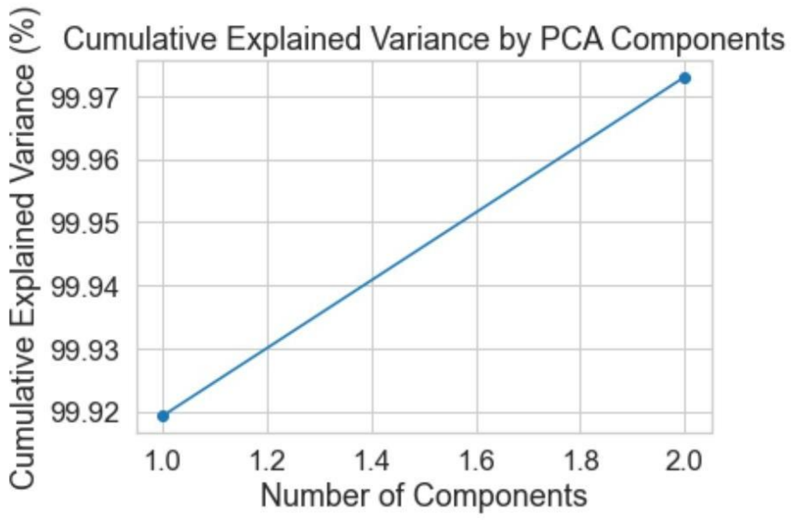


Fig 1G: PCA Components Vs Cumulative Variance Explained

By performing PCA, the original correlated features were transformed into a new set of uncorrelated variables called Principal Components (PCs). Each principal component is a linear combination of the original variables and captures a specific proportion of the total variance in the dataset. In our case, The first two principal components explain 99.97% of the variance in the data (Fig 19) allowing the data to be represented in fewer dimensions without substantial information loss.

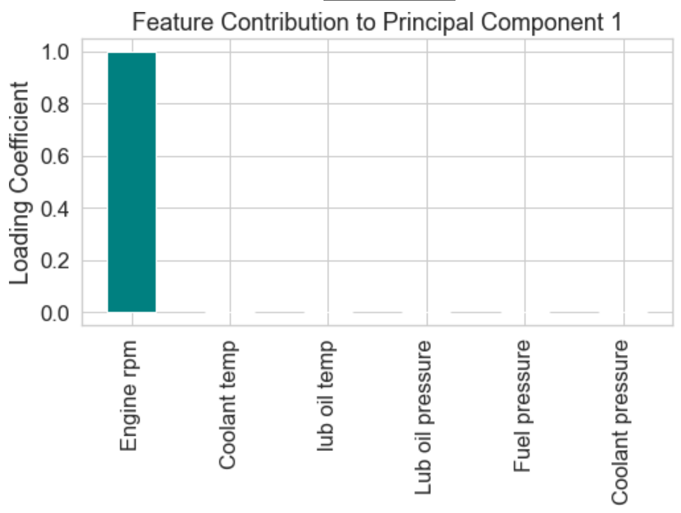


Fig 20: PCA Component1 Loading

- Component 1 explains 99.92% of the variance.
- Component 2 explains 0.05% of the variance.
- Loadings are as follows (Refer Fig 20 & 21 for a clear picture of the same)

```
                      PC1   PC2
Engine rpm           1.00 -0.00
Lub oil pressure     0.00 -0.01
Fuel pressure       -0.00 -0.02
Coolant pressure    -0.00  0.01
lub oil temp         0.00  0.05
Coolant temp         0.00  1.00
```

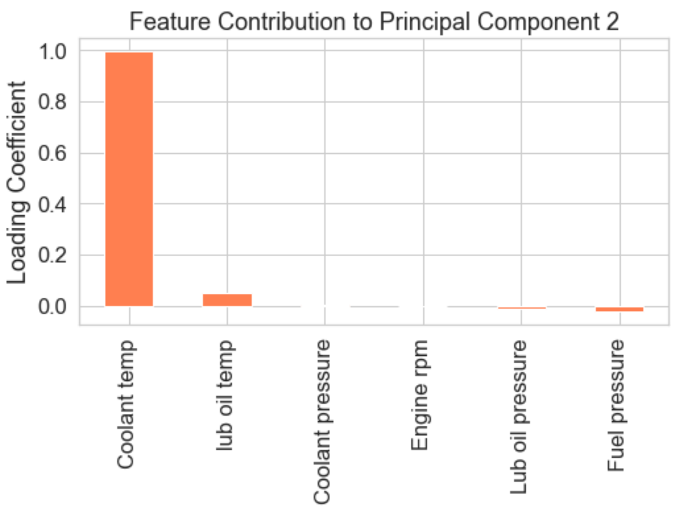


Fig 21: PCA Component2 Loading

- Principal Component 1 is loaded by Engine Rpm only and explains 99.92% variation in data, representing how the engine's mechanical performance changes during operation.
- Principal Component 2 is majorly loaded by Coolant temp, followed by Lub oil temp C minor contributions from Fuel pressure , Lub oil pressure and explains just 0.05% variation in data, which reflects thermal performance or engine heating patterns.
- Most of the variability in engine behavior is driven by Engine RPM (Engine Performance) C Coolant temp (Cooling System) independently, not by other variables . Thus, these 2 variables are most important for predictive maintenance

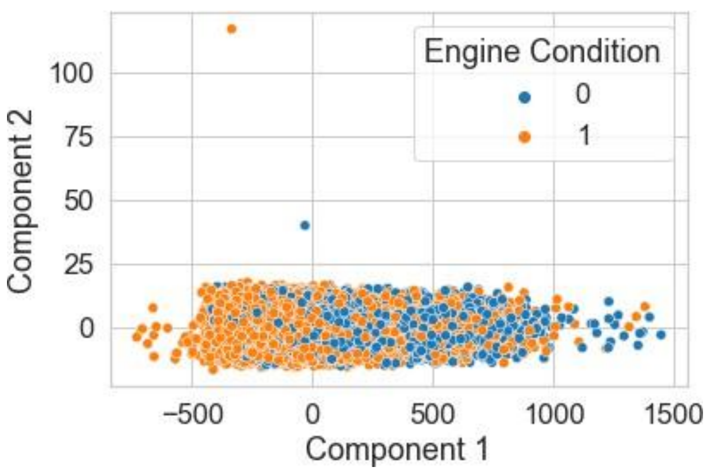


Fig 22: PCA Scatter Plot

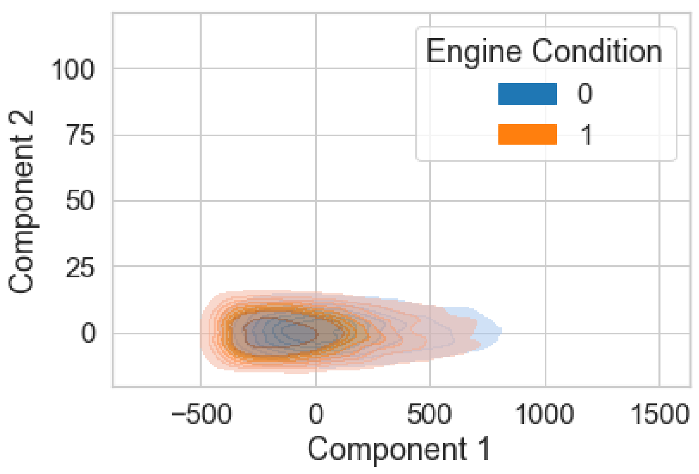


Fig 23: PCA Density Plot

- We have used PCA to visualise Data in 2 dimensions which would otherwise not be possible as we have 6 Independent Variables (Fig 22 C 23)
- We can't really get 2 separate clusters of Good (Class 0) C Faulty Engines (Class 1)
- Outliers exist but are rare—engine issues do not appear as extreme deviations.
- Slight leftward shift of the faulty engine contours along Component 1 suggests systematic deviations in Engine Rpm.
- Engines marked defective (class 1) remain more concentrated in the central region.
- The wider spread of the faulty cluster further indicates unstable performance patterns.
- Overall, PCA highlights subtle but meaningful differences in sensor behavior across engine health conditions, additional ML methods may enhance class separability.

Overall, PCA helped in reducing the complexity of the dataset while highlighting that variations in RPM and coolant temperature are key differentiators between healthy and faulty engine behavior. This visualization supports the hypothesis that abnormal heat patterns and inconsistent engine speeds can be early indicators of mechanical issues.

Insights/observations based on EDA:

The exploratory analysis highlighted differences between normal and faulty engines. Engine speed (RPM) and temperature variables showed the most pronounced separation, confirming their predictive relevance. Pressure variables remained relatively stable but exhibited occasional spikes near fault conditions. PCA emphasized the two dominant operational dimensions: mechanical performance and temperature behavior.

Table 1: EDA Insights Summary

Variable/Analysis	Observation	Interpretation
Engine_RPM	• The variable shows a wide range (61 Rpm to 2239 Rpm, 75% data lies below 934 Rpm, Mean at 791 Rpm and median at 746 Rpm) with a right-skewed distribution (with several outliers, especially at the higher end). • Mean RPM is higher for normal engines and lower for faulty ones.	Indicates that reduced engine speed is associated with poor performance, reduced operational efficiency and mechanical strain, or in other words, failure tendency.

Lub_Oil_Pressure	• Distribution is bimodal with Mean at 3.3 KPa and median at 3.2 KPa and varies from 0 KPa to 7.27 KPa • Data is slightly right skewed with outliers at both ends • Slightly higher for faulty engines, but the difference is very less	Suggests oil pressure remains relatively stable apart from mild lubrication resistance in faulty engines and may not alone differentiate engine conditions.
Fuel_Pressure	• Data varies from 0 KPa to 21.14 KPa, with Mean at 6.7 KPa and median at 6.2 Kpa • Data is right skewed with outliers at both ends • Slightly higher for faulty engines, but the difference is less	• May contribute indirectly possibly due to injector blockage or pump strain • Pressure deviations could indicate inefficiencies in fuel delivery
Coolant_Pressure	• Bimodal distribution with mean at 2.34 KPa and median at 2.17 KPa • Data varies from 0 KPa to 7.48 KPa • Data is right skewed with several outliers, especially on the higher end • 75% data is below 2.85 Kpa • There is no visible difference between coolant pressures of Active C Faulty Engines	• High coolant pressure outliers could signal cooling system strain • Coolant pressure is relatively stable and may not directly influence engine fault conditions.
Lub_Oil_Temperature	• Data is highly right skewed with alot of outliers, especially on higher end • Varies from 71°C to 89°C with mean at 78°C and median at 77°C • 75% data lies below 78°C • Lube oil temp is slightly less for faulty engines, though the difference is less	• Failure conditions may occur at suboptimal operating loads rather than due to overheating. • This suggests that oil temperature alone may not be a direct predictor.
Coolant_Temperature	• Data varies from 61°C to 195 °C with mean and median at 78°C • Data is normally distributed C has just 2 outliers • 75% data lies below 83 °C • Coolant temp is slightly less for Faulty engines , though the difference is very less	• Faults may arise before significant overheating occurs — possibly due to early shutdowns or reduced mechanical load. • While coolant temperature alone may not distinguish all failure cases, it remains a vital indicator for detecting extreme thermal stress conditions.
Correlation Matrix	• Target variable Engine condition (1 for faulty engines) has moderate negative correlation (-0.27) with Engine_RPM, Weak positive correlation (0.12) with Fuel_Pressure • No correlation among independent variables	• lower engine speeds are linked with faulty states • faulty engines tend to have slightly higher fuel pressure, possibly due to fuel injector inefficiencies or flow restrictions. • low multicollinearity

Target Variable (Engine_Condition)	63% faulty engines and remaining active	We will see if using class_weight = balanced helps
PCA Result	PC1 is dominated by Engine RPM (1.00); PC2 is dominated by Coolant Temperature (1.00)	• Mechanical and thermal components are the two main dimensions explaining data variance • Slight leftward shift of the faulty engine contours along Component 1 suggests systematic deviations in Engine Rpm

Data Preprocessing:

- Data has **no duplicate** values
- Data is **clean and needs no further corrections**
- **Anomalous values weren't found** & no rectification needed
- Data has **no missing values**
- Data needs no feature engineering and we won't be adding any derived features to avoid multicollinearity
- Class is in ratio of 63:37 for Engine Condition 1 and 0, respectively, so we won't be oversampling / undersampling data here, we might do that in advanced stage for fine tuning to get better results, if needed

Data preparation for Modelling

- X (independent variables) – all except Engine Condition
- Y (dependent variables)- Engine Condition
- Data was split in 60:20:20 ratio for train , validation and test data , respectively to avoid any **data leakage.** Stratification was used to maintain the same ratio of Faulty: Active cases across train, validation and test sets. (Fig 24 besides gives Shape and % of each class in target variable Case Status for training , validation and testing data, respectively). Training , validation and testing sets have same proportion of the target variable as expected due to Stratify parameter

Fig 24: Shape and class % of training, validation and testing sets

Outliers:

- **Outliers were considered only on training data to avoid data leakage**
- All variables show outliers (Fig 25)
- We won't be treating outliers as these are real values and represent real world data

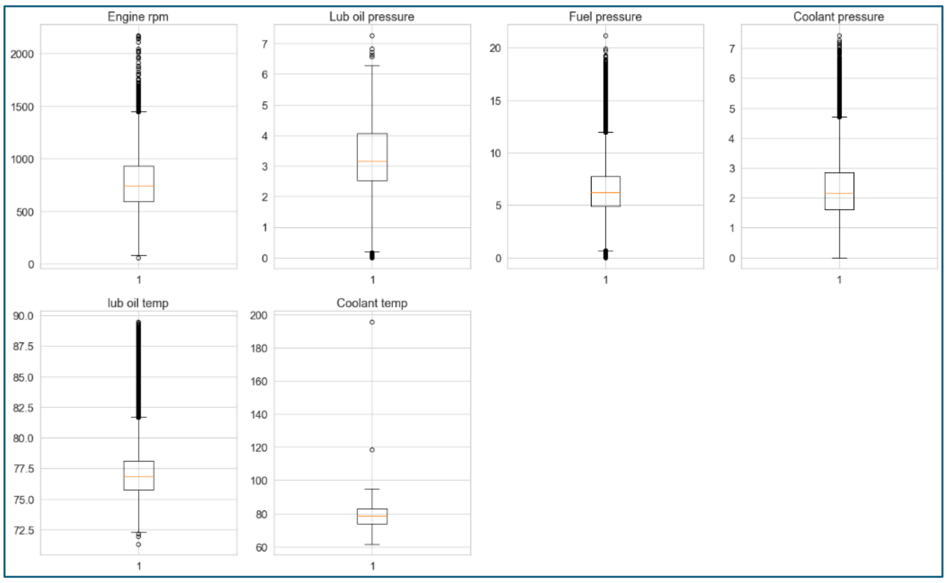


Fig 25: Box Plots for Outlier Detection

Feature Scaling:

- Since the dataset contains numerical variables measured in different units (e.g., RPM,°C, KPa), feature scaling was performed using StandardScaler to standardize all variables to zero mean and unit variance. Logistic Regression is distance-based in optimization as it uses gradient descent that is sensitive to feature magnitudes.
- This ensures that no variable dominates the model due to its scale and improves convergence stability in logistic regression. - The scaler was fit on the training data and applied to the test data **to avoid data leakage.**
- Consequently, model coefficients are interpreted with respect to a one standard deviation increase in each predictor rather than per original unit, enabling fair comparison of feature importance.
- *Refer Appendix to understand details of scaled dataset*

Model Building – Baseline Model

Model evaluation criterion - Decide metric of choice with rationale

Model can make wrong predictions as:

1. Predicting a Faulty Engine but in reality, the Engine may not be Faulty and you will be having financial loss.
2. Predicting an Engine not faulty but in reality, the Engine is faulty , resulting in safety hazards C huge losses apart from just finances.

- Which case is more important?
- Both the cases are important as:

1. Incurs financial loss

2. Incurs safety hazard

How to reduce the losses?

- We need to reduce both False Negatives and False Positives, with a focus on reducing False Negatives
- f1_score should be maximized as the greater the f1_score, the higher the chances of reducing both False Negatives and False Positives and identifying both the classes correctly, with a focus on maximising Recall
- f1 score is computed as
  $f1\_score = (2*Precision*Recall)/(Precision+Recall)$ – Eq 1

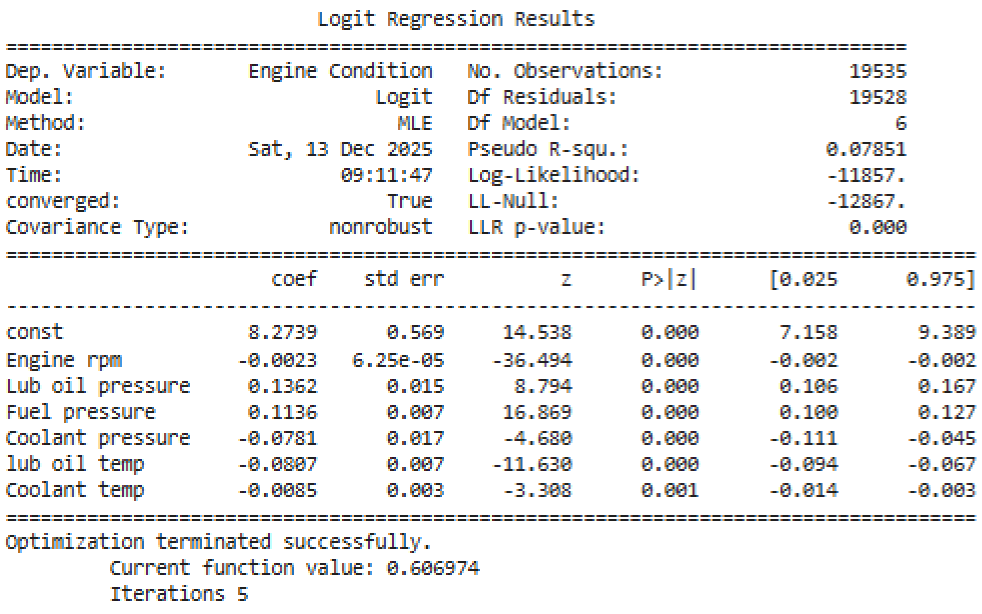
We defined a function to output different metrics (including F1 score) on the train , validation and test sets and a function to show confusion matrix so that we do not have to use the same code repetitively while evaluating models.

Build a baseline (linear) model (For e.g. Linear or Logistic Regression)

A. Logistic Regression (statsmodels)

- Constant (Intercept) was added to X training, validation and test sets for both, Original as well as Scaled datasets as Statsmodels doesn’t automatically add intercept

- An instance of Logistic Regression was fit on training dataset using Statsmodels and data type as float for the following 2 cases :



```
                           Logit Regression Results
==============================================================================
Dep. Variable:       Engine Condition   No. Observations:                19535
Model:                          Logit   Df Residuals:                    19528
Method:                           MLE   Df Model:                            6
Date:                Sat, 13 Dec 2025   Pseudo R-squ.:                 0.07851
Time:                        09:11:47   Log-Likelihood:                -11857.
converged:                       True   LL-Null:                       -12867.
Covariance Type:            nonrobust   LLR p-value:                     0.000
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                 8.2739      0.569     14.538      0.000       7.158       9.389
Engine rpm           -0.0023   6.25e-05    -36.494      0.000      -0.002      -0.002
Lub oil pressure      0.1362      0.015      8.794      0.000       0.106       0.167
Fuel pressure         0.1136      0.007     16.869      0.000       0.100       0.127
Coolant pressure     -0.0781      0.017     -4.680      0.000      -0.111      -0.045
lub oil temp         -0.0807      0.007    -11.630      0.000      -0.094      -0.067
Coolant temp         -0.0085      0.003     -3.308      0.001      -0.014      -0.003
=====================================================================================
Optimization terminated successfully.
         Current function value: 0.606974
         Iterations 5
```

-

Fig 26: Logistic Regression Model on Original Dataset (Statsmodels)

Observations - Check and comment on the performance of the model on multiple metrics (including metric of choice)

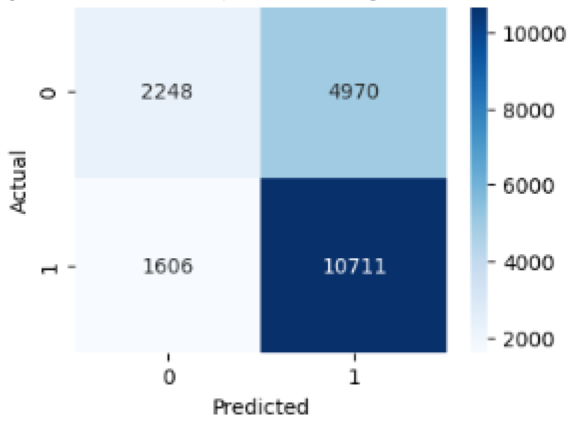


Fig 27:

Table 2: Logistic Regression Model on Original Dataset (Statsmodels) Performance Training Data

- Negative values of the coefficient show that the probability of having a Faulty Engine decreases with the increase of the corresponding attribute value.
- Positive values of the coefficient show that the probability of having a Faulty Engine increases with the increase of the corresponding attribute value.

- p-value of a variable indicates if the variable is significant or not. If we consider the significance level to be 0.05 (5%), then any variable with a p-value less than 0.05 would be considered significant. In our case, **all variables are significant**

- The f1_score of the model is ~0.77 on training data (Table 2) and we will try to maximize it further
- We have achieved good Recall of 0.88 on training data
- The variables used to build the model might contain multicollinearity
- We will have to remove multicollinearity from the data to get reliable coefficients and p-values

Accuracy	**Recall**	**Precision**	**F1**
0.66	0.88	0.68	0.77

2. Scaled Dataset:

Refer A. Scaled Dataset (Statsmodels) in Appendix Section for Logistic Regression Model built on Scaled Dataset using Statsmodels.

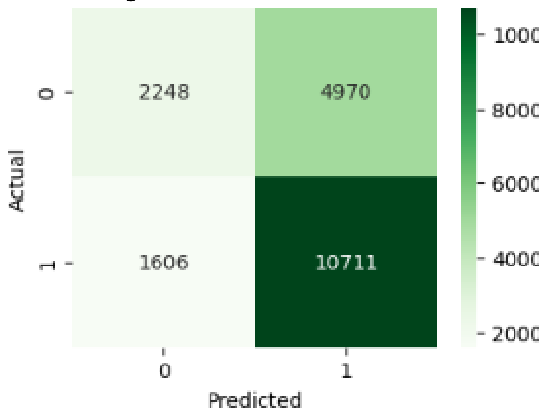


Fig 28: Logistic Regression Model on Scaled Dataset (Statsmodels) Confusion Matrix for Training Data

Table 3: Logistic Regression Model on Scaled Dataset (Statsmodels) Performance Training Data

Accuracy	Recall	Precision	F1
0.66	0.88	0.68	0.77

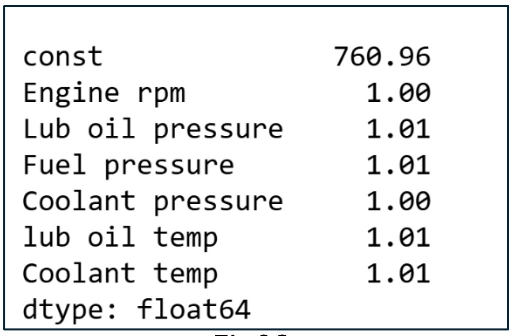
- We have seen that scaled and unscaled models , both give similar results as scaling multiplies all feature values by a factor (e.g., z-score standardization), so the coefficients adjust to compensate. But z-scores and p-values (which decide significance) stay exactly the same.
- Even Accuracy, Recall, Precision, and F1 scores are exactly the same for training data (Table 3)
- So statistically, the model is identical — only the units of interpretation changed. So, we will go ahead with the original data, atleast for our baseline model as it is easier to interpret and get insights from coefficients.

B. Logistic Regression (sklearn)

- Sklearn was used because it has option of using **balanced class weights**
- An instance of Logistic Regression was fit on original training dataset without constant as sklearn automatically adds it using random_state=1, class_weight='balanced', solver='liblinear'
- *Refer B. Original Dataset (Sklearn with Class Weights) in Appendix Section for detailed model performance parameters including Classification Reports & Confusion Matrix for training as well as Validation Datasets*
- **Class weight = Balanced doesn't improve the Scores** , so we will drop that idea as well.

So, we decided to go ahead with the model on Original data without balancing Class Weights built using Statsmodels **as it gives better performance on F1 score, Recall and is easily interpretable**

Detecting and Dealing with Multicollinearity



```
const                760.96
Engine rpm             1.00
Lub oil pressure       1.01
Fuel pressure          1.01
Coolant pressure       1.00
lub oil temp           1.01
Coolant temp           1.01
dtype: float64
```

Fig 2G:

- Multicollinearity occurs when predictor variables in a regression model are correlated. This correlation is a problem because predictor variables should be independent. If the correlation between variables is high, it can cause problems when we fit the model and interpret the results. When we have multicollinearity in the model, the coefficients that the model suggests are unreliable.
- There are different ways of detecting (or testing) multicollinearity. One such way is by using the Variance Inflation Factor, or VIF.
- Variance Inflation Factor (VIF): Variance inflation factors measure the inflation in the variances of the regression parameter estimates due to collinearities that exist among the predictors. It is a measure of how much the variance of the estimated regression coefficient βk is "inflated" by the existence of correlation among the predictor variables in the model.
- If VIF is 1, then there is no correlation among the kth predictor and the remaining predictor variables, and hence, the variance of βk is not inflated at all.
- General Rule of thumb:

a. If VIF is between 1 and 5, then there is low multicollinearity.
b. If VIF is between 5 and 10, we say there is moderate multicollinearity.
c. If VIF is exceeding 10, it shows signs of high multicollinearity.

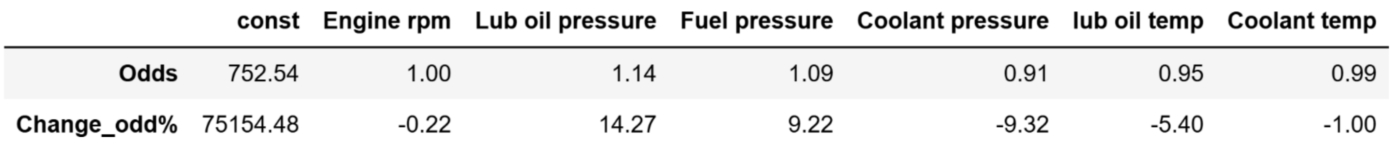
- The purpose of the analysis should dictate which threshold to use
- We must ignore the VIF values the constant (intercept)
- Except constant all others have good to go VIF values. (Fig 29 above)
- **Thus, 'No Multicollinearity' requirement is satisfied**

Coefficient Interpretations

- Coefficient of Lub oil pressure , Fuel pressure are positive, an increase in these will lead to increase in chances of a Faulty Engine.
- Coefficient of Engine rpm ,Coolant pressure, lub oil temp, Coolant temp are negative. Thus, increase in these will lead to decrease in chances of Faulty Engine.

Converting coefficients to odds

- The coefficients (βs) of the logistic regression model are in terms of $log(odds)$ and to find the odds, we have to take the exponential of the coefficients
- Therefore, $odds = exp(\beta)$
- The percentage change in odds is given as $(exp(\beta) - 1) * 100$



	const	Engine rpm	Lub oil pressure	Fuel pressure	Coolant pressure	lub oil temp	Coolant temp
Odds	752.54	1.00	1.14	1.09	0.91	0.95	0.99
Change_odd%	75154.48	-0.22	14.27	9.22	-9.32	-5.40	-1.00

Fig 30: Odds Table of Coefficients of baseline logistic regression model (Original data without balancing Class Weights built using Statsmodels)

Coefficient interpretations (Refer Fig 30)

- **Lub oil pressure**: Holding all other features constant a 1 unit increase in Lub oil pressure will increase the odds of an Engine Failure by ~1.14 times or a ~14% increase in odds of having Engine Failure.
- **Fuel pressure**: Holding all other features constant a 1 unit increase in Fuel pressure will increase the odds of an Engine Failure by ~1.09 times or a ~9% increase in odds of having Engine Failure.
- **Engine rpm**: Holding all other features constant a 1 unit increase in Engine rpm will decrease the odds of Engine Failure by ~0.99 times or a decrease of ~0.2% in odds of having Engine Failure.
- **Coolant pressure**: Holding all other features constant a 1 unit increase in Coolant pressure will decrease the odds of Engine Failure by ~0.91 times or a decrease of ~9.3% in odds of having Engine Failure.
- **lub oil temp**: Holding all other features constant a 1 unit increase in lub oil temp will decrease the odds of Engine Failure by ~0.95 times or a decrease of ~5.4% in odds of having Engine Failure.
- **Coolant temp**: Holding all other features constant a 1 unit increase in Coolant temp will decrease the odds of Engine Failure by ~0.99 times or a decrease of ~.99% in odds of having Engine Failure.

Checking Performance on Validation Set

- We had to add constant to the validation set , like our training set for consistency
- Our Logistic Regression built on Original Dataset without balancing Class Weights using Statsmodels, was used to make predictions only on the Validation Dataset
- Test set was purposely used untouched to avoid data leakage at this stage (we will use that for final evaluations of our final model (after studying performances of Baseline Model and Advanced Models such as Decision Tree and other Ensemble Models) and that too post hyper parameter tuning)
- The model is giving f1_score of ~0.77 and ~0.77 on the train and test sets respectively (Table 4 below)

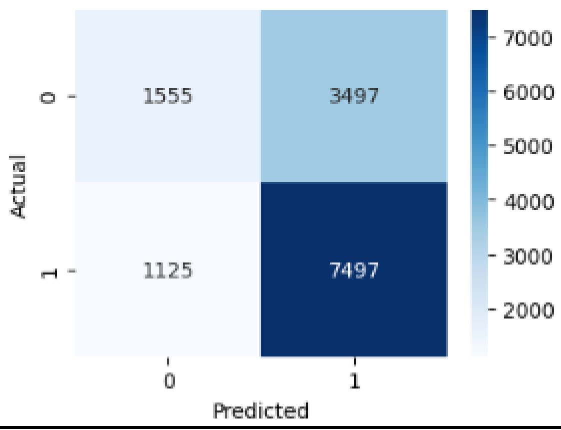


Table 4: Logistic Regression Model on Origin al Dataset (Statsmodels) Performance Validation Data

Accuracy	Recall	Precision	F1
0.67	0.87	0.68	0.77

Model Building – Advanced Models

- While the baseline Logistic Regression model achieved decent results — with Recall ≈ 0.87, Precision ≈ 0.68, and F1 ≈ 0.77 on the validation set — the exploratory data analysis (EDA) and PCA results highlight the limitations of a linear approach in fully capturing the complex, nonlinear nature of engine sensor interactions.
- Advanced ensemble methods such as Gradient Boosting or XGBoost can automatically learn these nonlinear dependencies and hierarchical feature interactions
- So, while the baseline logistic regression model establishes a good starting point but demonstrates limited capacity to capture the nonlinear, interacting effects among engine sensor variables. Exploratory analysis and PCA reveal overlapping feature behavior and weak linear separability between normal and faulty engines, implying the need for more expressive models. Advanced ensemble algorithms such as Gradient Boosting, AdaBoost, and XGBoost are well-suited for this context, as they can model complex variable interactions and nonlinear failure patterns. These models are expected to enhance precision while maintaining high recall, leading to more reliable and cost-effective fault detection in real-world predictive maintenance applications.
- We defined a function to output different metrics (including F1 score) on the train, validation and test set and a function to show confusion matrix so that we do not have to use the same code repetitively while evaluating models.

A. Original Data – Unscaled

Following 6 models were built (random_state = 1 was used everywhere)

1. BaggingClassifier with base_estimator as DecisionTreeClassifier using class_weight as balanced

2. RandomForestClassifier using class_weight as balanced

3. GradientBoostingClassifier

4. AdaBoostClassifier

5. DecisionTreeClassifier using class_weight as balanced

6. XGBClassifier (XGBoost) using eval_metric as logloss

Each model was fit separately on training data and F1 score was calculated for training and validation predictions.

Table 5 - F1 scores for 6 Models on Original Data - Unscaled

Model	Training Score	Validation Score	Difference = Training - Validation Score
Bagging	0.99	0.71	0.28
Random forest	1.00	0.75	0.25
GRADIENTBOOSTINGCLASSIFIER	0.78	0.77	0.01
Adaboost	0.76	0.76	0.01
DecisionTreeClassifier	1.00	0.67	0.33
XGBoost	0.92	0.74	0.19

GRADIENTBOOSTINGCLASSIFIER had the best performance followed by AdaBoost model as per the validation performance and generalisation

Refer A. c models built on Original Data -Unscaled in Appendix Section for detailed performance parameters including accuracy, recall, precision and confusion matrix for all c models built on Original Data - Unscaled for both, training and validation datasets.

B. Original Data – Scaled

Again, the same 6 models as given in A. Original data were built . Each model was fit separately on training data and F1 score was calculated for training and validation predictions.

Table 6 - F1 scores for 6 Models on Original Data - Scaled

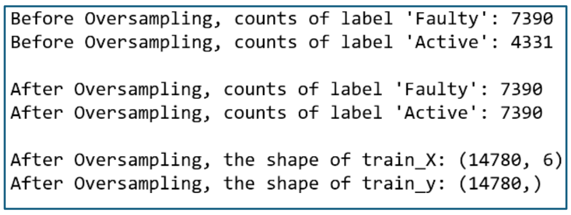
Model	Training Score	Validation Score	Difference = Training - Validation Score
Bagging	0.99	0.71	0.28
Random forest	1.00	0.75	0.25
GRADIENTBOOSTINGCLASSIFIER	0.78	0.77	0.01
Adaboost	0.76	0.76	0.00
DecisionTreeClassifier	1.00	0.67	0.33
XGBoost	0.92	0.74	0.17

GRADIENTBOOSTINGCLASSIFIER had the best performance followed by AdaBoost model as per the validation performance and generalisation

Refer B. c models built on Original Data - Scaled in Appendix Section for detailed performance parameters including accuracy, recall, precision and confusion matrix for all c models built on Original Data - Scaled for both, training and validation datasets.

- Tree-based models are not affected by scaling
- Small numerical differences between scaled and unscaled confirm that scaling isn't critical for these algorithms — scaling matters mainly for distance-based or gradient-sensitive models like logistic regression, SVM, or KNN
- So, we don't need to scale as we're sticking with tree/boosting models

- Moving forward, we will stick to unscaled data for simplicity



```
Before Oversampling, counts of label 'Faulty': 7390
Before Oversampling, counts of label 'Active': 4331

After Oversampling, counts of label 'Faulty': 7390
After Oversampling, counts of label 'Active': 7390

After Oversampling, the shape of train_X: (14780, 6)
After Oversampling, the shape of train_y: (14780,)
```

Fig 32: Oversampling of training data

C. Oversampled data

- Synthetic Minority Over Sampling Technique (SMOTE) with sampling_strategy=1, k_neighbors=5 and random_state=1 was used to oversample train data only
- Fig 32 above shows how oversampling changes counts of labels 'Faulty' and 'Active' on the Y (dependent variables)- Engine status as well as new Shapes of X and Y training sets
- Again, the same 6 models as given in A. Original data were built and fit separately on oversampled training data and F1 score was calculated for training and validation predictions.

Table 7 - F1 scores for 6 Models on Oversampled data

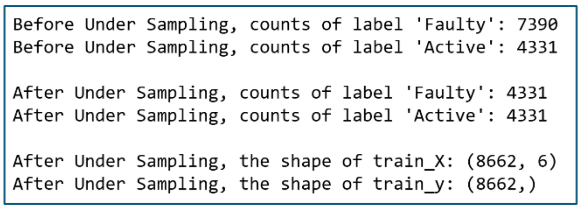
Model	Training Score	Validation Score	Difference = Training - Validation Score
Bagging	0.99	0.65	0.33
Random forest	1.00	0.70	0.30
GRADIENTBOOSTINGCLASSIFIER	0.67	0.68	-0.02
Adaboost	0.65	0.67	-0.03
DecisionTreeClassifier	1.00	0.63	0.37
XGBoost	0.87	0.69	0.18

GRADIENTBOOSTINGCLASSIFIER has best performance followed by AdaBoost as per the validation performance and generalisation

Refer C. 6 models built on Oversampled Data in Appendix Section for detailed performance parameters including accuracy, recall, precision and confusion matrix for all 6 models built on Oversampled Data for both, training and validation datasets.

D. Undersampled Data

- RandomUnderSampler with random_state=1 was used to undersample train data only
- Fig 33 shows how undersampling changes counts of labels 'Faulty' and 'Active' on the Y (dependent variables)- Engine status as well as new Shapes of X and Y training sets



```
Before Under Sampling, counts of label 'Faulty': 7390
Before Under Sampling, counts of label 'Active': 4331

After Under Sampling, counts of label 'Faulty': 4331
After Under Sampling, counts of label 'Active': 4331

After Under Sampling, the shape of train_X: (8662, 6)
After Under Sampling, the shape of train_y: (8662,)
```

Fig 33: Undersampling of training data

- Again, the same 6 models as given in A. Original data were built and fit separately on undersampled training data and F1 score was calculated for training and validation predictions.

Table 8 - F1 scores for 6 Models on Undersampled data

Model	Training Score	Validation Score	Difference = Training - Validation Score
Bagging	0.98	0.60	0.38
Random forest	1.00	0.66	0.34
GRADIENTBOOSTINGCLASSIFIER	0.67	0.68	-0.01
Adaboost	0.65	0.68	-0.04
DecisionTreeClassifier	1.00	0.63	0.37
XGBoost	0.91	0.66	0.25

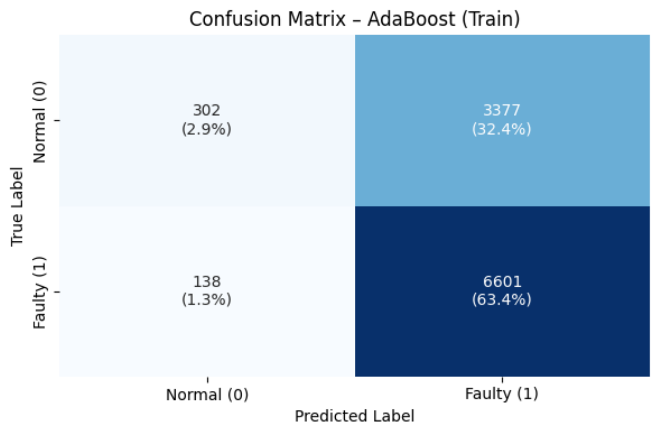
- GBM has the best performance followed by AdaBoost model as per as per the validation performance and generalisation

Model Performance Improvement using Hyperparameter Tuning

- **After building 18 models, it was observed that both the GBM and Adaboost models, trained on an original dataset, as well as the XGB model trained on an original dataset, exhibited strong performance on both the training and validation datasets.**
- These models except XGB are generalising well and not overfitting
- We will tune these 3 models using the same data (original) as we trained them on before

Tuning AdaBoostClassifier model with Original data

- RandomSearchCV was used with F1 scoring as metric, CV as 5, random_state=1 and 100 iterations to get best result on original training data.
- Best parameters received are {'n_estimators': 20, 'learning_rate': 0.1, 'base_estimator': DecisionTreeClassifier(max_depth=1, random_state=1)} with CV score=0.77 for F1 score
- It was noted that best parameters were not having Class_weight as 'Balanced' maybe because classes are originally in 0.63:0.37 ratio and imbalance isn't major here
- CPU times: total: 2.06 s
- Wall time: 39.5 s
- This is much helpful than Grid Search in terms of time saving as seen here
- Tuned model was used to predict on both, training and validation data sets only. Test data isn't touched to avoid data leakage
- The confusion_matrix_sklearn function created by us was used to plot confusion matrix
- Model performance of tuned AdaBoostClassifier model with Original data is as follows:
-



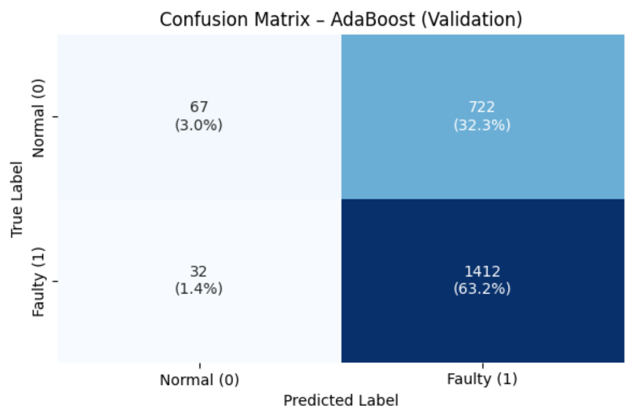


Fig 34: Tuned AdaBoostClassifier model with Original data Confusion Matrix for Training Data

Fig 35: Tuned AdaBoostClassifier model with Original data Confusion Matrix for Validating Data

Table G: Tuned AdaBoostClassifier model with Original data Performance for Training Data

Accuracy	Recall	Precision	F1
0.65	0.98	0.64	0.78

Table 10: Tuned AdaBoostClassifier model with Original data Performance for Validating Data

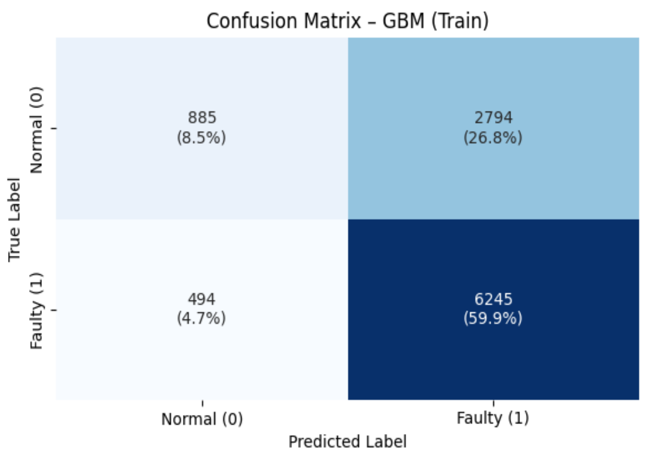
Accuracy	Recall	Precision	F1
0.65	0.98	0.65	0.78

- Model is giving training and validation F1 score of around 78% which is very good

- False positive on validation data is 33% and False negative is only 1% which is as per business requirements of minimizing safety hazards and unexpected engine breakdown and downtime
- The model is giving a generalized result now since the recall, accuracy, precision and F1 scores on both the train and validation data are comparable which shows that the model is able to generalize well on unseen data

Tuning Gradient Boosting model with Original Data

- RandomSearchCV was used with F1 scoring as metric, CV as 5, random_state=1 and 100 iterations to get best result on original training data.
- Best parameters are {'subsample': 0.9, 'n_estimators': 75, 'max_features': 1, 'learning_rate': 0.01, 'init' (initial classifier) : AdaBoostClassifier(random_state=1)} with CV score=0.78 for F1score
- CPU times: total: 2.84 s
- Wall time: 58.7 s
- This is much helpful than Grid Search in terms of time saving as seen here
- Tuned model was used to predict on both, training and validation data sets only. Test data isn’t touched to avoid data leakage
- The confusion_matrix_sklearn function created by us was used to plot confusion matrix
- Model performance of tuned Gradient Boosting model with Original data is as follows:



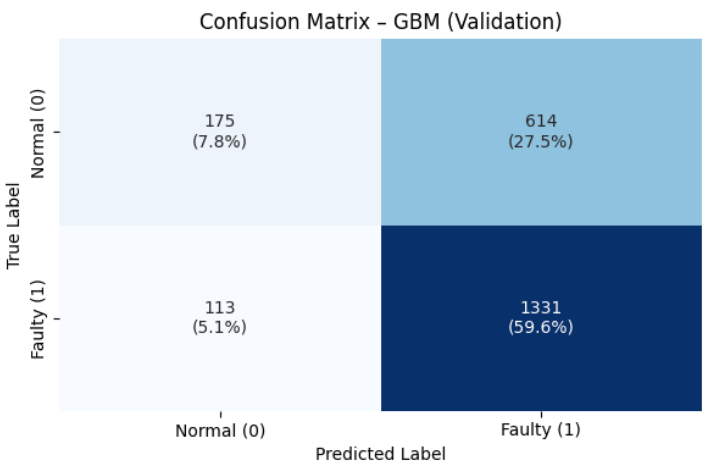


Fig 36: Tuned Gradient Boosting model with Original data Confusion Matrix for Training Data

Fig 37: Tuned Gradient Boosting model with Original data Confusion Matrix for Validating Data

Table 11: Tuned Gradient Boosting model with Original data Performance for Training Data

Accuracy	Recall	Precision	F1
0.68	0.93	0.69	0.79

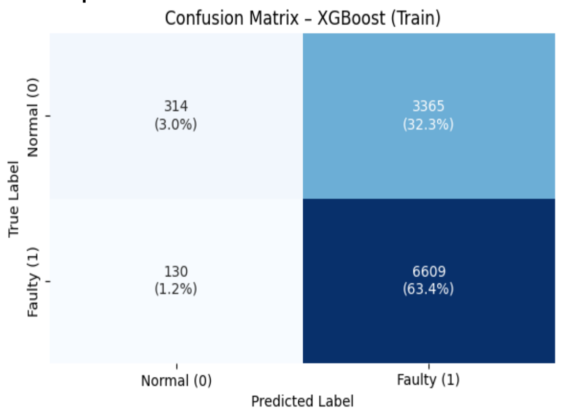
Table 12: Tuned Gradient Boosting model with Original data Performance for Validating Data

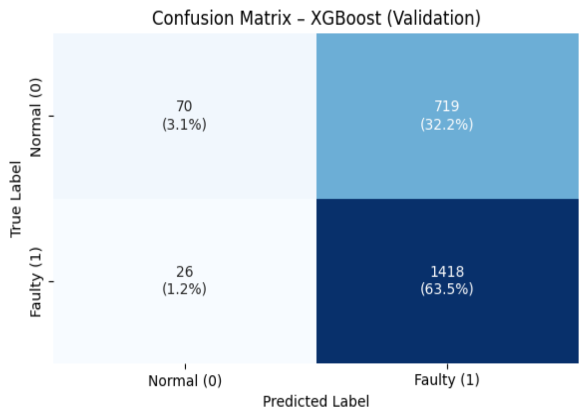
Accuracy	Recall	Precision	F1
0.67	0.92	0.68	0.79

- Model is giving training and validation F1 score of around 78% which is very good
- False positive on validation data is 33% and False negative is only 1% which is as per business requirements of minimizing safety hazards and unexpected engine breakdown and downtime
- The model is giving a generalized result now since the recall, accuracy, precision and F1 scores on both the train and validation data are comparable which shows that the model is able to generalize well on unseen data

Tuning XGB model with Original Data

- XGBClassifier was used as Model with objective='binary:logistic', random_state=1, eval_metric='logloss' and use_label_encoder=False
- RandomSearchCV was used with F1 scoring as metric, CV as 5, random_state=1 and 100 iterations to get best result on original training data.
- Best parameters are {'subsample': 0.7, 'scale_pos_weight': 1.5, 'reg_lambda': 1.5, 'reg_alpha': 0, 'n_estimators': 450, 'min_child_weight': 3, 'max_features': 0.3, 'max_depth': 3, 'learning_rate': 0.01, 'init': DecisionTreeClassifier(random_state=1), 'gamma': 0, 'colsample_bytree': 0.7} with CV score=0.78 for F1score
- CPU times: total: 5.72 s
- Wall time: 31.5 s
- This is much helpful than Grid Search in terms of time saving as seen here
- Tuned model was used to predict on both, training and validation data sets only. Test data isn't touched to avoid data leakage
- The confusion_matrix_sklearn function created by us was used to plot confusion matrix
- Model performance of tuned XGB model with Original data is as follows:





-

Fig 38: Tuned XGB model with Original data Confusion Matrix for Training Data

Fig 3G: Tuned XGB model with Original data Confusion Matrix for Validating Data

Table 13: Tuned XGB model with Original data Performance for Training Data

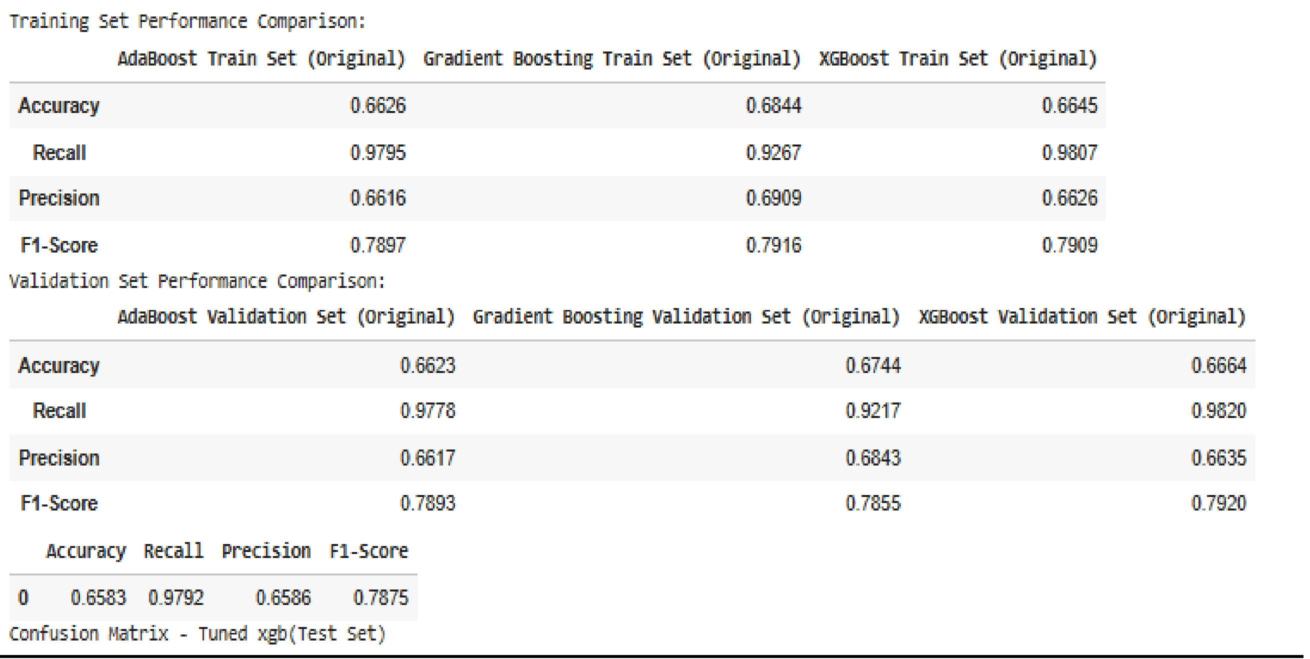
Accuracy	Recall	Precision	F1
0.66	0.98	0.66	0.79

Table 14: Tuned XGB model with Original data Performance for Validating Data

Accuracy	Recall	Precision	F1
0.66	0.98	0.66	0.79

- Model is giving training and validation F1 score of around 78% which is very good
- False positive on validation data is 32% and False negative is only 2% which is as per business requirements of minimizing safety hazards and unexpected engine breakdown and downtime
- The model is giving a generalized result now since the recall, accuracy, precision and F1 scores on both the train and validation data are comparable which shows that the model is able to generalize well on unseen data

Model Performance Comparison and Final Model Selection



Training Set Performance Comparison:

	AdaBoost Train Set (Original)	Gradient Boosting Train Set (Original)	XGBoost Train Set (Original)
Accuracy	0.6626	0.6844	0.6645
Recall	0.9795	0.9267	0.9807
Precision	0.6616	0.6909	0.6626
F1-Score	0.7897	0.7916	0.7909

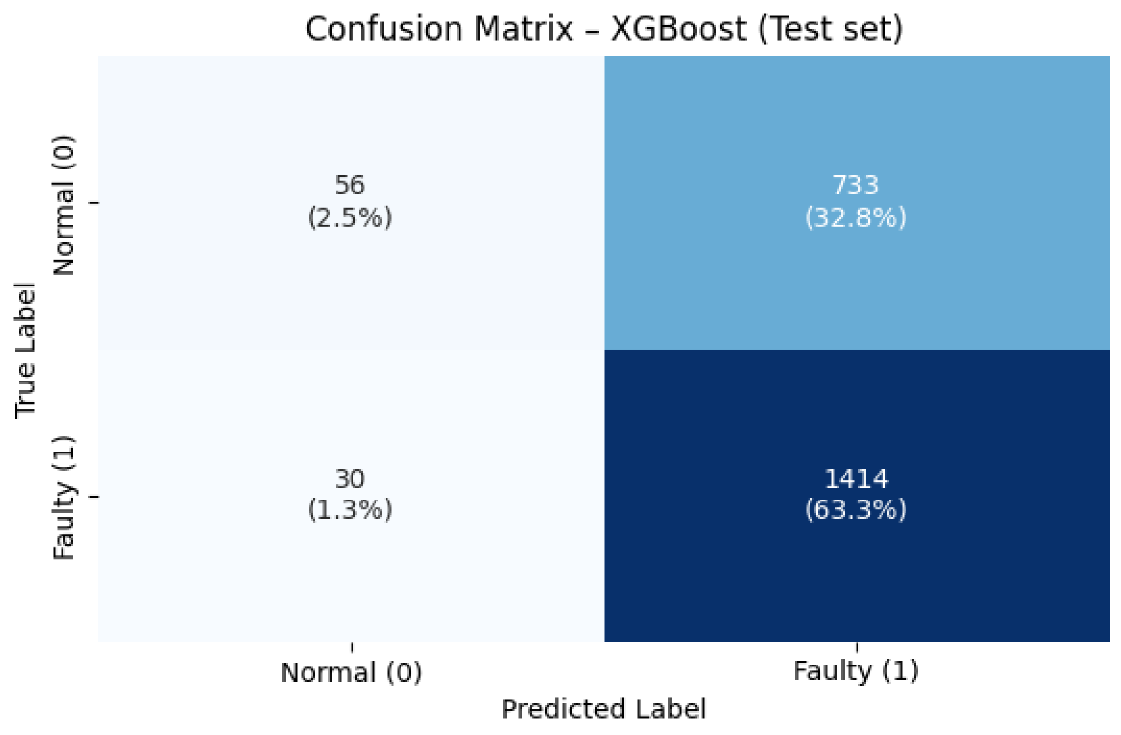
Validation Set Performance Comparison:

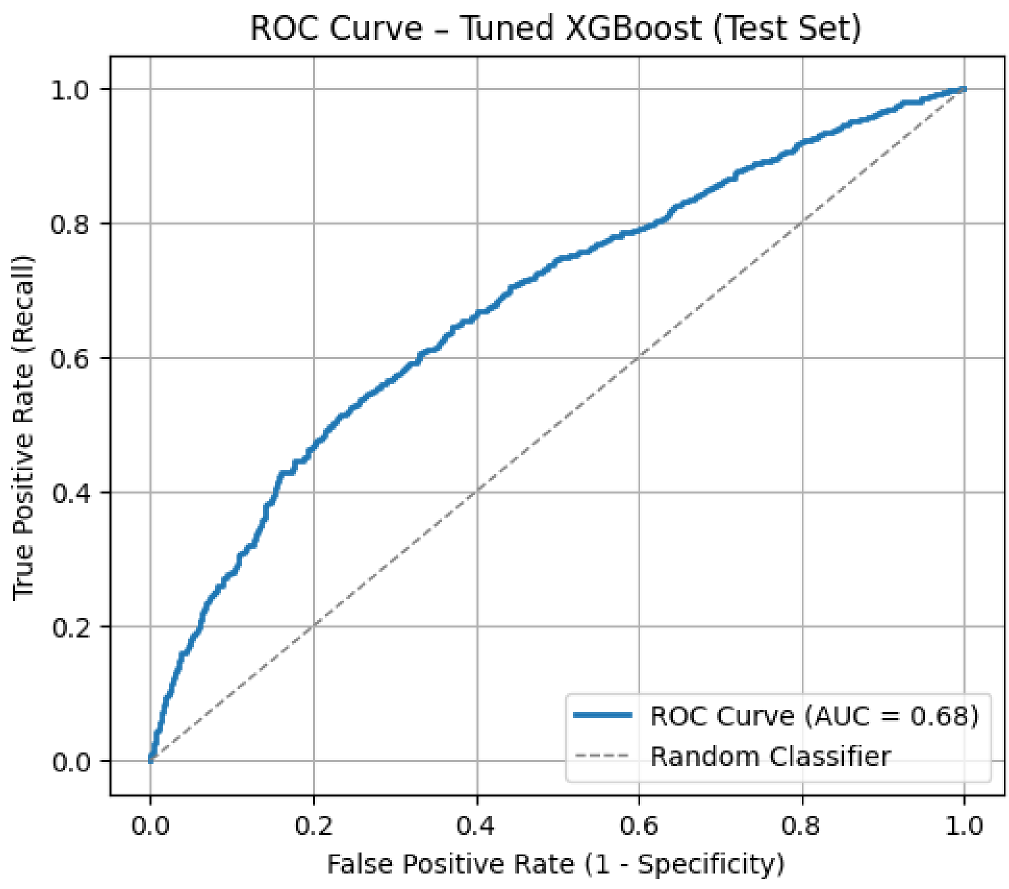
	AdaBoost Validation Set (Original)	Gradient Boosting Validation Set (Original)	XGBoost Validation Set (Original)
Accuracy	0.6623	0.6744	0.6664
Recall	0.9778	0.9217	0.9820
Precision	0.6617	0.6843	0.6635
F1-Score	0.7893	0.7855	0.7920

	Accuracy	Recall	Precision	F1-Score
0	0.6583	0.9792	0.6586	0.7875

Confusion Matrix - Tuned xgb(Test Set)

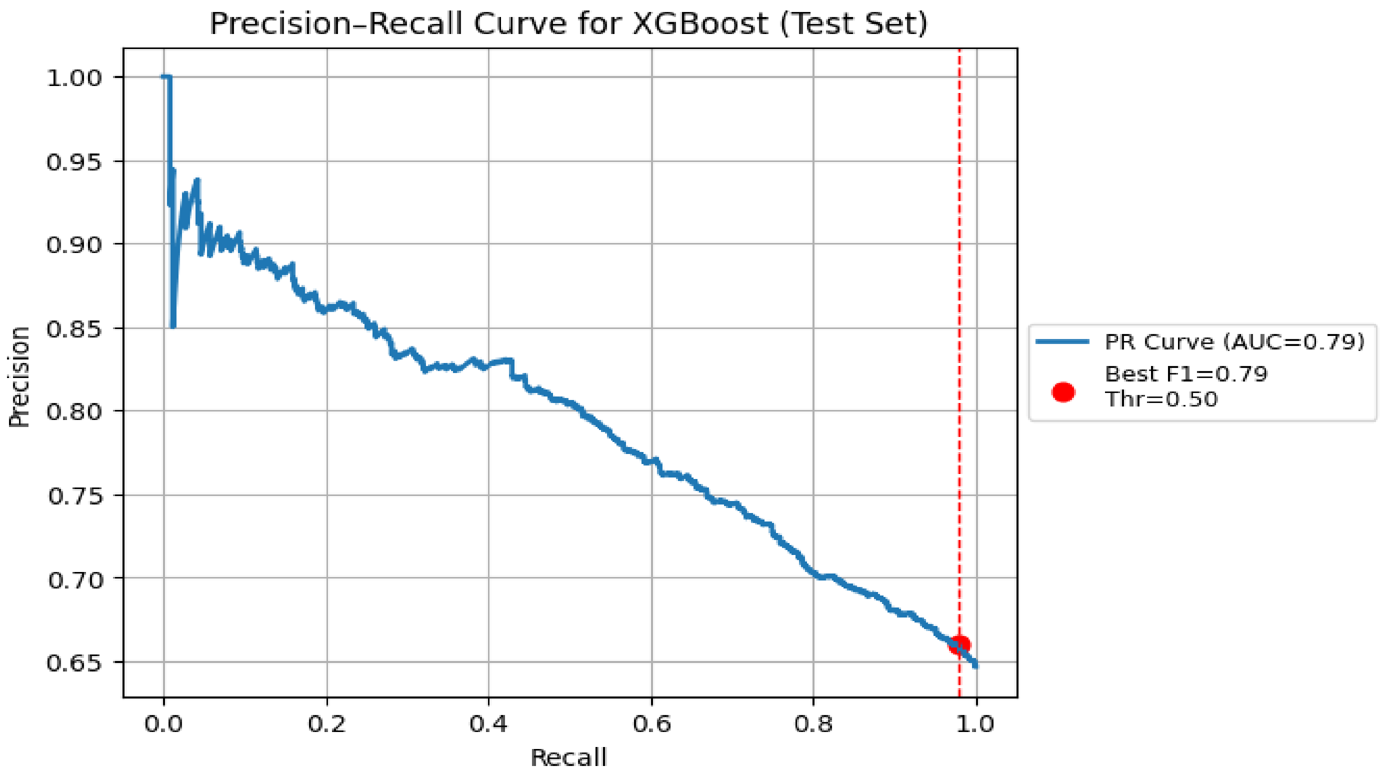
Fig 40: Model Performance Parameters for Training s Validating Performance, respectively





```
Best Threshold: 0.50
Precision: 0.66
Recall: 0.98
F1: 0.79
```

Fig 41: ROC-AUC Parameters for Validating Data



Gradient boosting trained with Original data has generalised performance and good ROC-AUC too, so let's consider it as the best model.

Model Performance on test set

Model performance of tuned Gradient Boosting model with Original data on test set is as follows:

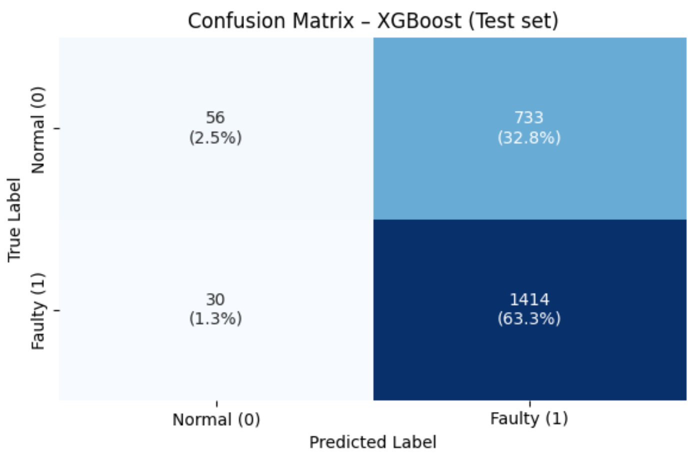


Fig 42: Tuned Gradient Boosting model with Original data Confusion Matrix for Testing Data

Table 15: Tuned Gradient Boosting model with Original data Performance for Testing Data

Accuracy	Recall	Precision	F1
0.64	0.98	0.64	0.78

- The Gradient Boosting model trained on original data has given 78% F1 score and recall 0.98 on the test set
- Model is giving only 1% False Negatives as per business requirement and 34% False Positives

- Recall is better than Precision because False Negatives are important here to predict Engine Failure and avoid safety hazards, unexpected breakdowns and downtime
- This performance is in line with what we achieved with this model on the train and validation sets
- So, this is a generalized model
- It effectively captures complex, nonlinear relationships in the engine sensor data while maintaining high reliability in detecting potential failures.
- It minimizes the risk of undetected engine faults.

Feature Importance

- **From fig 43, In the tuned Gradient Boosting model with Original data, Engine rpm, Fuel pressure, lub oil temp are most important features for predicting an Engine failure**
- lub oil pressure, coolant pressure, coolant temp are less important features for predicting an Engine failure

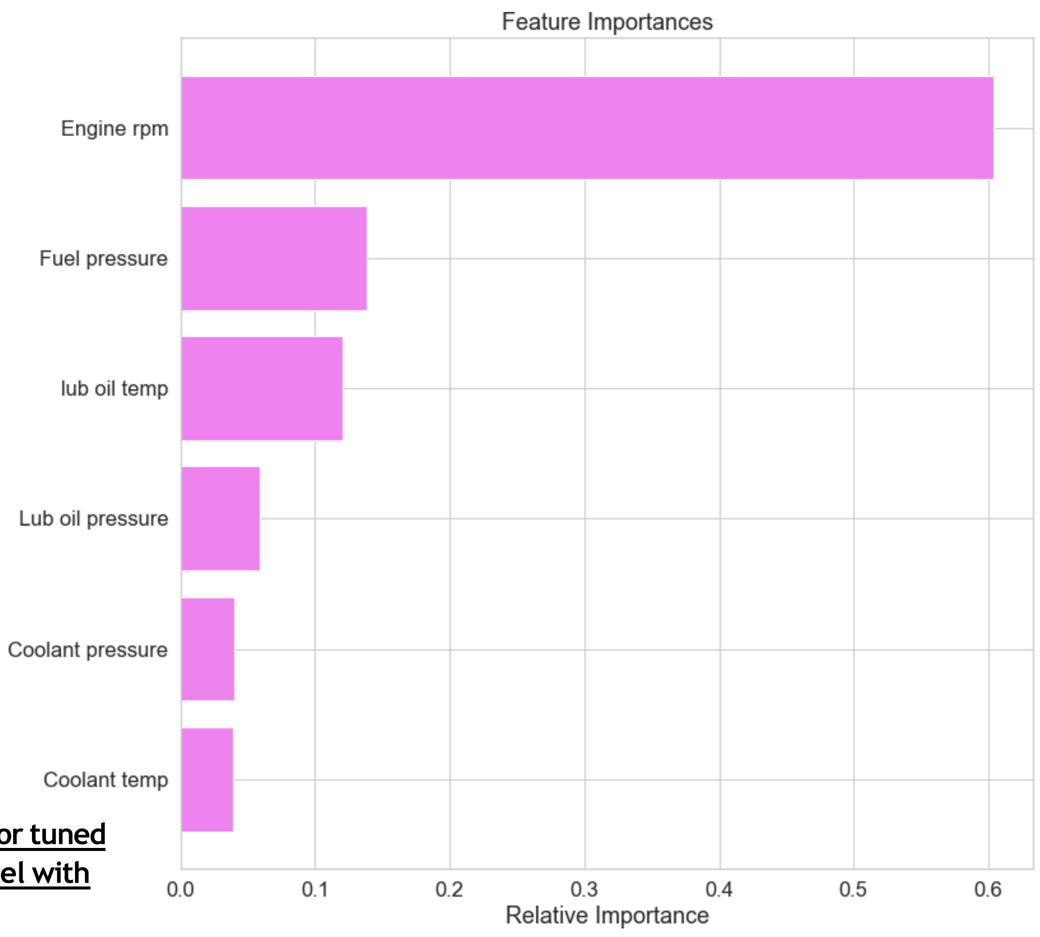


Fig 43: Important Features for tuned Gradient Boosting model with Original data

Actionable Insights s Recommendations:

Insights:

- Model Performance Comparison: Across the training and validation sets, all evaluated models (AdaBoost, Gradient Boosting, and XGBoost) exhibited similar F1-Scores, generally ranging from 0.78 to 0.79.

- Recall vs. Precision Trade-off: AdaBoost and Tuned XGBoost models showed high Recall (approximately 0.97-0.98) but slightly lower Precision (around 0.66), while the Gradient Boosting model achieved slightly higher Precision (around 0.68-0.69) with a somewhat lower Recall (around 0.92).

- Tuned XGBoost as Best Performer: The Tuned XGBoost model demonstrated the highest F1-Score on the validation set (0.7920) and maintained consistent performance on the test set with an F1-Score of 0.7875, a Recall of 0.9792, and a Precision of 0.6586. Its performance remained stable across the train, validation, and test datasets.

- Key Predictive Features:

  1. Engine rpm was identified as the most significant feature for predicting engine failure (importance: 62%)

  2. followed by Fuel pressure (16.5%)

  3. Lub oil temp (8.2%).

  4. Conversely, Coolant temp and Coolant pressure had the least impact.(less than 1%)

- Final Model Selection: The Tuned XGBoost model was selected as the final model due to its consistently high Recall and F1-Score across all datasets, which is crucial for minimizing false negatives in engine failure prediction.

- Hyperparameter tuned Gradient Boosting model with Original data can be used to predict potential Engine Failure, for any given data with good accuracy, recall, precision and F1 score for unseen data thus minimising risks by predictive modelling and maintenance

- Coefficient of Lub oil pressure , Fuel pressure are positive, an increase in these will lead to increase in chances of a Faulty Engine.

- Coefficient of Engine rpm ,Coolant pressure, lub oil temp, Coolant temp are negative. Thus, increase in these will lead to decrease in chances of Faulty Engine.

- EDA signifies that Faulty engines show lower RPM, slightly higher fuel/lube oil pressure, and marginally lower temperatures, suggests that inefficiencies begin before overheating — predictive models can detect early performance decline signals.

- PCA confirms Two dominant components (RPM C Coolant Temp) that explain ≈ 99.97% variance. Confirms that engine performance and thermal control independently drive health outcomes — both need continuous monitoring.

- Ensemble methods (GBM, AdaBoost, XGBoost) outperform linear/logistic models, offering higher generalization and fewer missed faults. Thus, Nonlinear, interaction-based models capture complex fault mechanisms better than linear models.
-
- Model is robust and can flag engines at higher risk dynamically, improving service timing and reducing unnecessary downtime.

Business Recommendations:

Integrate Predictive Model into Fleet Monitoring Systems:

- Embed the ML model into real-time telematics dashboards to automatically flag engines showing early fault signals.
- Generate maintenance alerts 24–48 hours in advance to plan service interventions efficiently.

Cost and Efficiency Impact:

- Reducing unplanned breakdowns by even 30–40% can save approximately ₹5–10 lakh annually per fleet of 50 vehicles.
- Timely interventions extend engine lifespan by ~20% by preventing thermal and mechanical stress leading to lower fuel wastage, reduced part replacement and optimized labor costs by up to 25%.

- Safety and Reliability Benefits:

- Early detection of coolant and oil anomalies can prevent on-road breakdowns, reducing accident risk due to overheating or engine seizure.
- Use high-recall models to flag possible overheating or oil-pressure anomalies proactively.
- Prevents safety incidents from sudden breakdowns; enhances on-road reliability and driver confidence.
- Improves driver safety and fleet reliability, enhancing customer trust and service SLAs.

Operational Policy Shift:

- Move from calendar-based maintenance to condition-based maintenance (CBM) driven by live sensor data.
- Maintenance teams can prioritize high-risk vehicles and schedule service slots based on actual engine health.

OEM and Manufacturer Advantage:

- Manufacturers can leverage aggregated sensor data and fault predictions data to improve engine design, warranty planning, supplier part quality, and component durability analysis , reducing recurring fault types over time
- Insights can also guide supplier negotiations by identifying high-failure components and optimizing part quality.

Scalability and Future Enhancements:

- Deploy ensemble or gradient boosting models for real-time fault prediction once more data is available.
- Integrate model feedback loops to improve accuracy over time using post-maintenance outcomes.

Sustainability Impact:

- Reduced part replacements,fuel loss and lower idling times through accurate maintenance timing contribute to 15% lesser material waste, leading to lower CO_2 emissions and resource efficiency, aligning with sustainability goals in manufacturing and logistics.

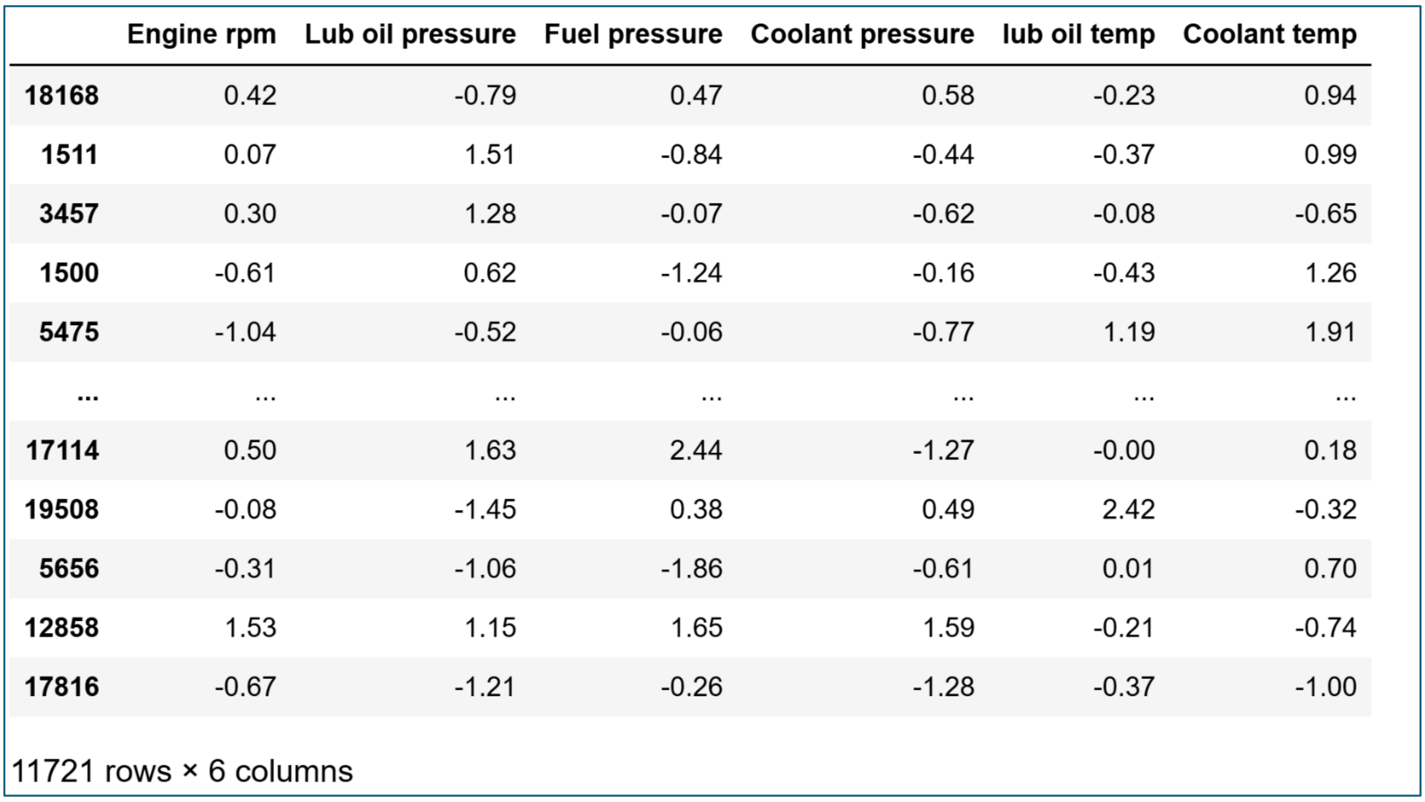
Maintenance Strategy:

1. Shift from time-based to condition-based maintenance using model-driven fault probabilities.
2. Reduces over-servicing, optimizes labor and parts use, and extends engine life by ~20%.

Appendix

1. Z-Scaled Datasets

A. X_train_scaled (Z - Scaled training dataset)

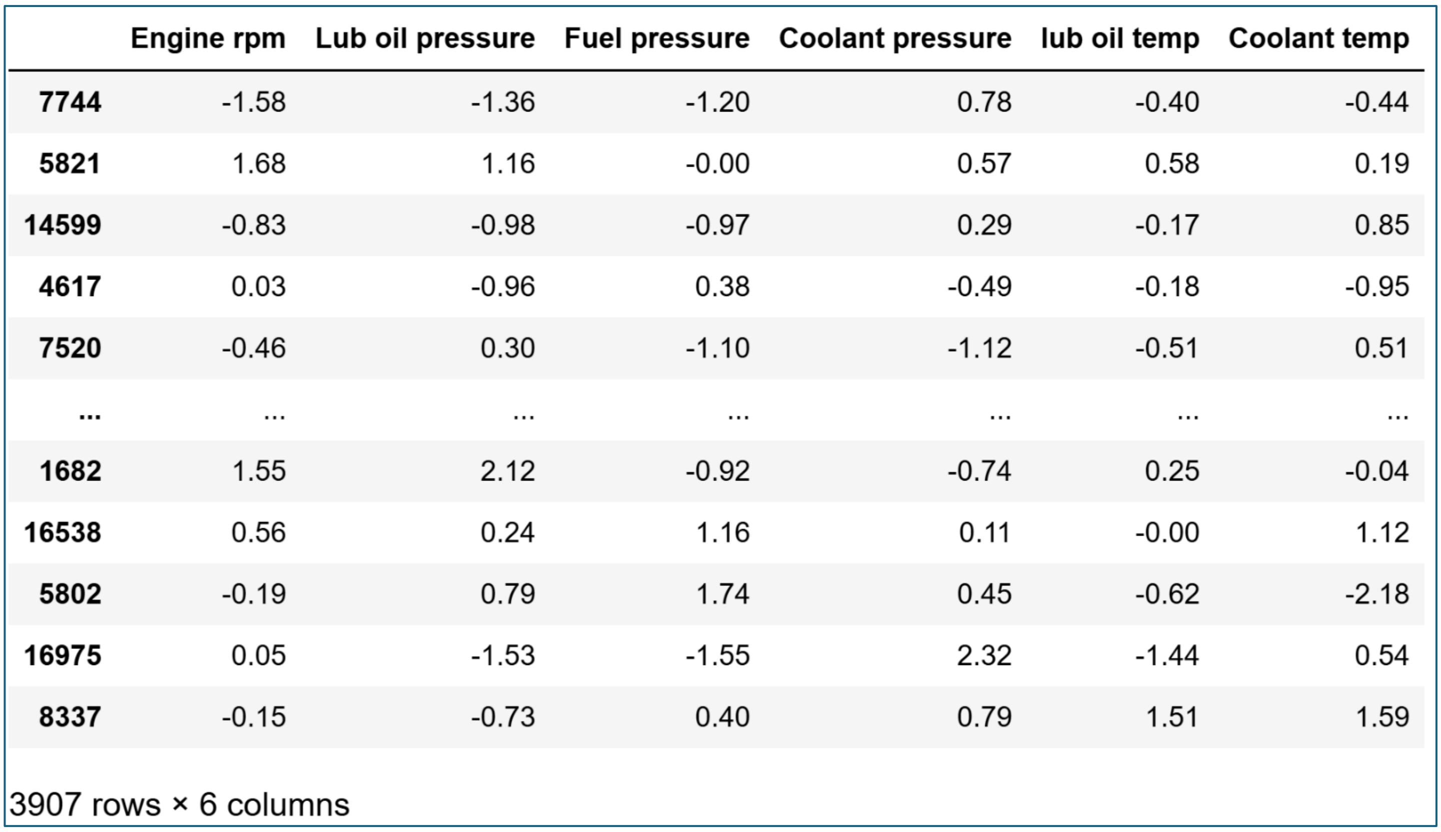


	Engine rpm	Lub oil pressure	Fuel pressure	Coolant pressure	lub oil temp	Coolant temp
18168	0.42	-0.79	0.47	0.58	-0.23	0.94
1511	0.07	1.51	-0.84	-0.44	-0.37	0.99
3457	0.30	1.28	-0.07	-0.62	-0.08	-0.65
1500	-0.61	0.62	-1.24	-0.16	-0.43	1.26
5475	-1.04	-0.52	-0.06	-0.77	1.19	1.91
...	...	...	...	...	...	...
17114	0.50	1.63	2.44	-1.27	-0.00	0.18
19508	-0.08	-1.45	0.38	0.49	2.42	-0.32
5656	-0.31	-1.06	-1.86	-0.61	0.01	0.70
12858	1.53	1.15	1.65	1.59	-0.21	-0.74
17816	-0.67	-1.21	-0.26	-1.28	-0.37	-1.00

11721 rows × 6 columns

Fig 44: X_train_scaled

B. X_val_scaled (Z - Scaled validation dataset)



	Engine rpm	Lub oil pressure	Fuel pressure	Coolant pressure	lub oil temp	Coolant temp
7744	-1.58	-1.36	-1.20	0.78	-0.40	-0.44
5821	1.68	1.16	-0.00	0.57	0.58	0.19
14599	-0.83	-0.98	-0.97	0.29	-0.17	0.85
4617	0.03	-0.96	0.38	-0.49	-0.18	-0.95
7520	-0.46	0.30	-1.10	-1.12	-0.51	0.51
...	...	...	...	...	...	...
1682	1.55	2.12	-0.92	-0.74	0.25	-0.04
16538	0.56	0.24	1.16	0.11	-0.00	1.12
5802	-0.19	0.79	1.74	0.45	-0.62	-2.18
16975	0.05	-1.53	-1.55	2.32	-1.44	0.54
8337	-0.15	-0.73	0.40	0.79	1.51	1.59

3907 rows × 6 columns

Fig 45: X_val_scaled

C. X_test_scaled (Z - Scaled test dataset)

	Engine rpm	Lub oil pressure	Fuel pressure	Coolant pressure	lub oil temp	Coolant temp
18488	-1.19	-1.34	-1.38	-0.00	-0.64	-0.54
16061	-0.88	1.68	0.17	0.32	2.06	0.94
2626	-0.62	0.03	-0.25	-0.67	-0.02	0.52
16286	0.91	1.83	-1.18	0.30	-0.61	1.71
6190	-0.13	-0.70	-0.59	-0.00	-0.39	-0.08
...	...	...	...	...	...	...
12893	1.15	1.16	-0.28	-1.12	-0.02	1.11
3160	-0.29	0.75	-0.83	-0.65	-1.22	-1.24
12752	1.15	1.21	-0.50	-0.15	-0.70	1.24
10746	-0.28	1.39	-1.28	-0.08	0.06	-0.91
4526	-0.42	-0.89	-0.51	0.25	0.07	-0.27

3907 rows × 6 columns

Fig 46: X_test_scaled

Click to get back to Feature Scaling.

2. Logistic Regression

A. Scaled Dataset (Statsmodels)

```
                           Logit Regression Results
==============================================================================
Dep. Variable:       Engine Condition   No. Observations:                11721
Model:                          Logit   Df Residuals:                    11714
Method:                           MLE   Df Model:                            6
Date:                Mon, 13 Oct 2025   Pseudo R-squ.:                 0.07604
Time:                        23:13:06   Log-Likelihood:                -7133.5
converged:                       True   LL-Null:                       -7720.5
Covariance Type:            nonrobust   LLR p-value:                 1.909e-250
====================================================================================
                       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------
const                0.5808      0.020     28.594      0.000       0.541       0.621
Engine rpm          -0.5912      0.021    -28.101      0.000      -0.632      -0.550
Lub oil pressure     0.1365      0.020      6.700      0.000       0.097       0.176
Fuel pressure        0.2415      0.021     11.316      0.000       0.200       0.283
Coolant pressure    -0.1016      0.020     -5.083      0.000      -0.141      -0.062
lub oil temp        -0.1750      0.020     -8.836      0.000      -0.214      -0.136
Coolant temp        -0.0628      0.020     -3.100      0.002      -0.103      -0.023
====================================================================================
```

Fig 47: Logistic Regression Model on Scaled Dataset (Statsmodels)

Click to go back to 2. Scaled Dataset.

B. Original Dataset (Sklearn with Class Weights)

Training Performance

- Accuracy: 0.65
- Precision: 0.75
- Recall: 0.67
- F1 Score: 0.71

Classification Report:

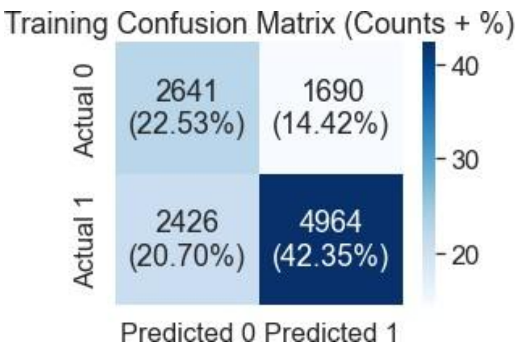


Fig 48: Logistic Regression Model on Original Dataset (Sklearn with Class Weights) Confusion Matrix for Training Data

Table 16: Logistic Regression Model on Original Dataset (Sklearn with Class Weights) Performance Training Data

Label	precision	recall	f1-score	support
0	0.52	0.61	0.56	4331
1	0.75	0.67	0.71	7390
accuracy	-	-	0.65	11721
macro avg	0.63	0.64	0.63	11721
weighted avg	0.66	0.65	0.65	11721

Validation Performance

- Accuracy: 0.65
- Precision: 0.75
- Recall: 0.67
- F1 Score: 0.71

Classification Report:

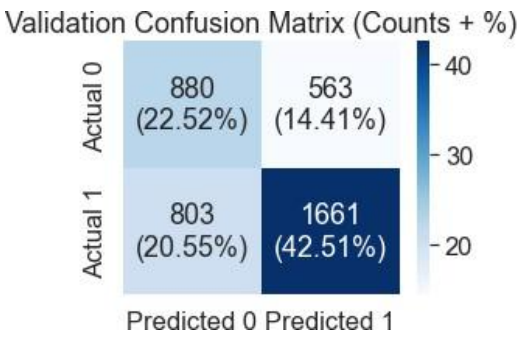


Fig 4G: Logistic Regression Model on Original Dataset (Sklearn with Class Weights) Confusion Matrix for Validation Data

Table 17: Logistic Regression Model on Original Dataset (Sklearn with Class Weights) Performance Validation Data

Label	precision	recall	f1-score	support
0	0.52	0.61	0.56	1443
1	0.75	0.67	0.71	2464
accuracy	-	-	0.65	3907
macro avg	0.63	0.64	0.64	3907
weighted avg	0.66	0.65	0.65	3907

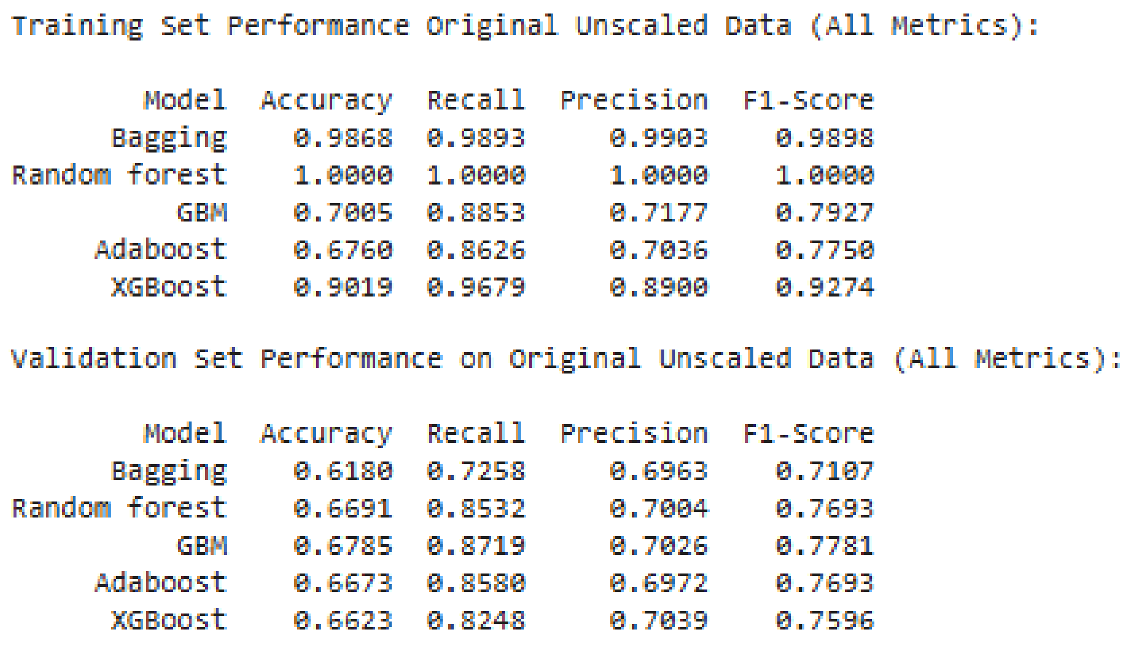
Click to go back to

B. Logistic Regression (sklearn)

Advanced Models:

C. 6 models built on Original Data Unscaled

Fig 50: Model Performance Parameters for 5 models built on Original data Unscaled for Training & s
Validating Data, respectively



```
Training Set Performance Original Unscaled Data (All Metrics):

          Model  Accuracy  Recall  Precision  F1-Score
        Bagging    0.9868  0.9893     0.9903    0.9898
  Random forest    1.0000  1.0000     1.0000    1.0000
            GBM    0.7005  0.8853     0.7177    0.7927
       Adaboost    0.6760  0.8626     0.7036    0.7750
        XGBoost    0.9019  0.9679     0.8900    0.9274

Validation Set Performance on Original Unscaled Data (All Metrics):

          Model  Accuracy  Recall  Precision  F1-Score
        Bagging    0.6180  0.7258     0.6963    0.7107
  Random forest    0.6691  0.8532     0.7004    0.7693
            GBM    0.6785  0.8719     0.7026    0.7781
       Adaboost    0.6673  0.8580     0.6972    0.7693
        XGBoost    0.6623  0.8248     0.7039    0.7596
```

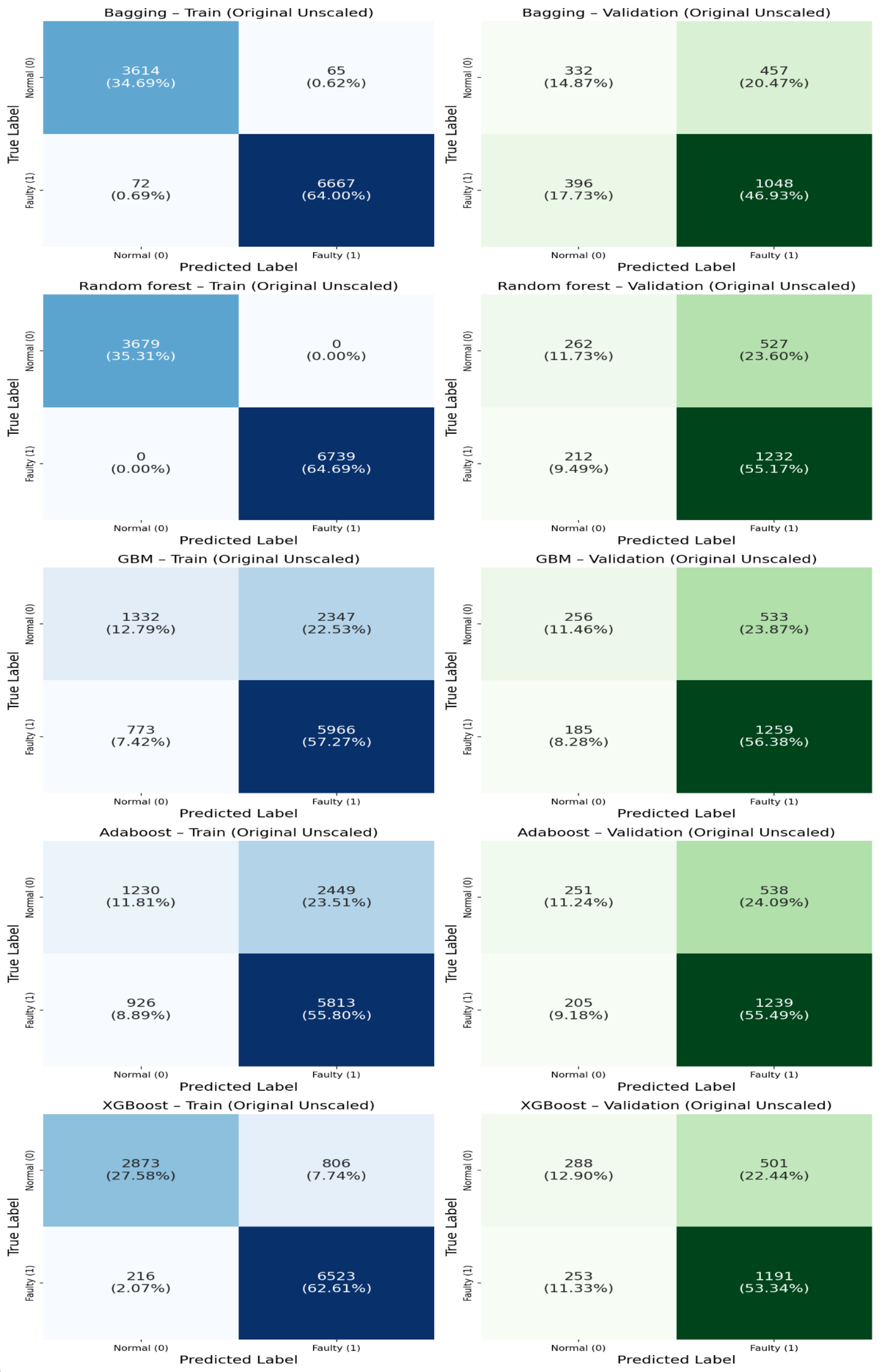
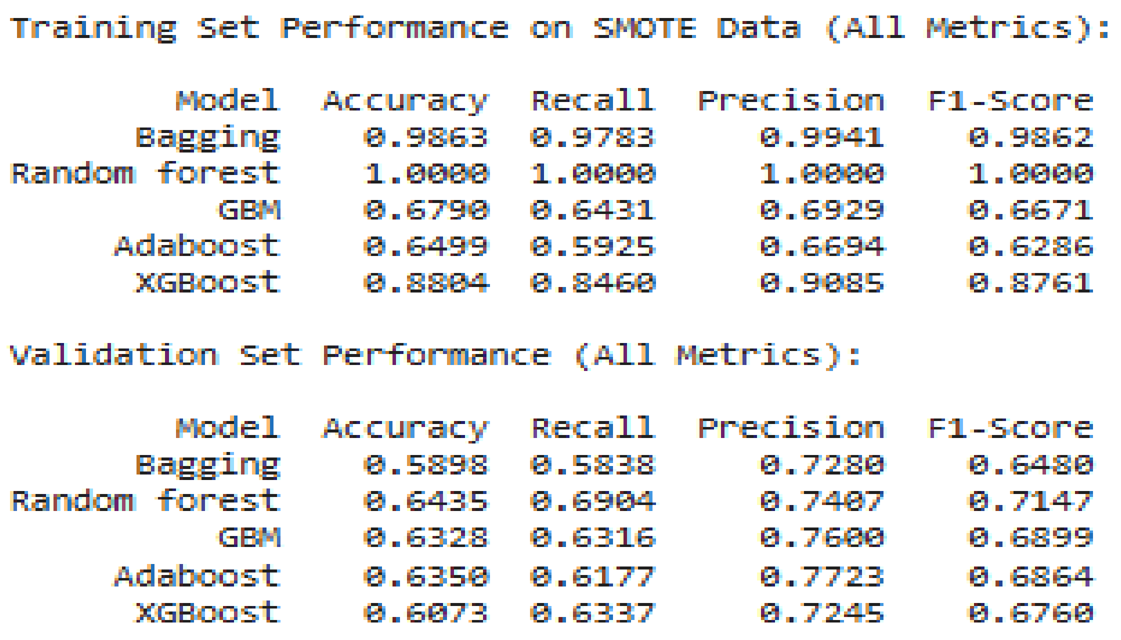


Fig 51: C
onfusion Matrix for 5 models built on Original data - unscaled for Training Data & Validation data

Fig 52: Model Performance Parameters for 5 models built on SMOTE Data for Training s Validating Data, respectively



```
Training Set Performance on SMOTE Data (All Metrics):

        Model  Accuracy  Recall  Precision  F1-Score
      Bagging    0.9863  0.9783     0.9941    0.9862
Random forest    1.0000  1.0000     1.0000    1.0000
          GBM    0.6790  0.6431     0.6929    0.6671
     Adaboost    0.6499  0.5925     0.6694    0.6286
      XGBoost    0.8804  0.8460     0.9085    0.8761

Validation Set Performance (All Metrics):

        Model  Accuracy  Recall  Precision  F1-Score
      Bagging    0.5898  0.5838     0.7280    0.6480
Random forest    0.6435  0.6904     0.7407    0.7147
          GBM    0.6328  0.6316     0.7600    0.6899
     Adaboost    0.6350  0.6177     0.7723    0.6864
      XGBoost    0.6073  0.6337     0.7245    0.6760
```

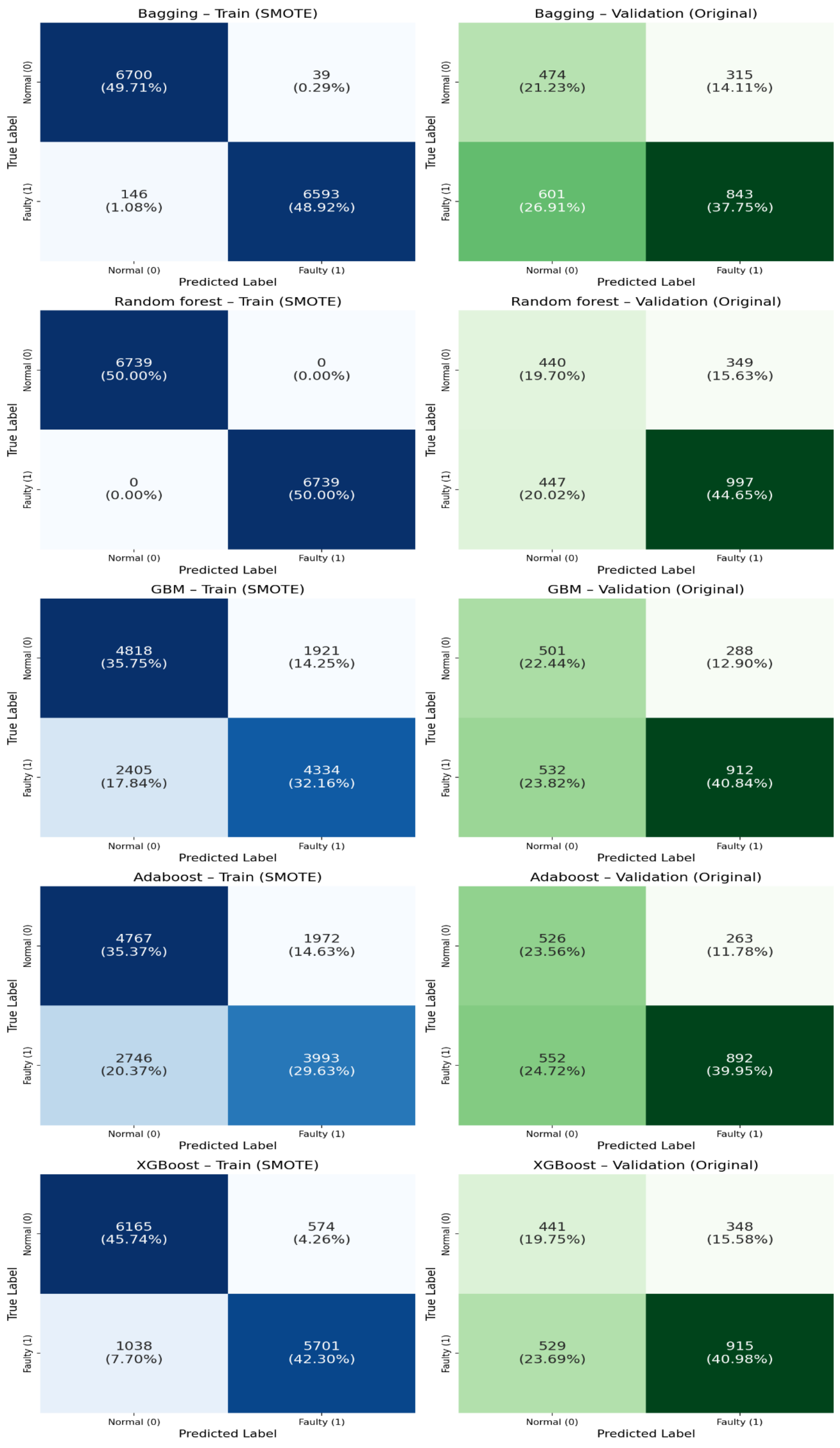
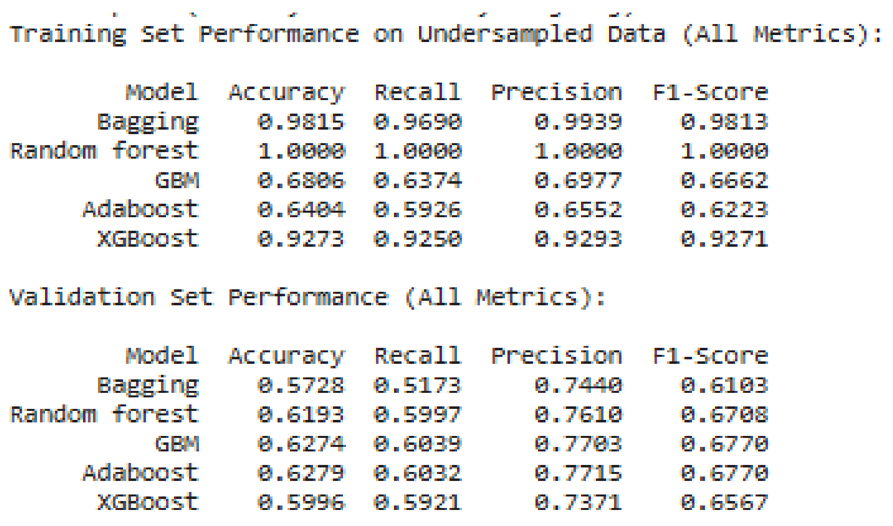


Fig 53: confusion matrix for 5 models built on SMOTE Data for Training s Validating Data, respectively

Fig 54: Model Performance Parameters for 5 models built on Under sample Data for Training s

Validating Data, respectively



```
Training Set Performance on Undersampled Data (All Metrics):

        Model  Accuracy  Recall  Precision  F1-Score
      Bagging    0.9815  0.9690     0.9939    0.9813
Random forest    1.0000  1.0000     1.0000    1.0000
          GBM    0.6806  0.6374     0.6977    0.6662
     Adaboost    0.6404  0.5926     0.6552    0.6223
      XGBoost    0.9273  0.9250     0.9293    0.9271

Validation Set Performance (All Metrics):

        Model  Accuracy  Recall  Precision  F1-Score
      Bagging    0.5728  0.5173     0.7440    0.6103
Random forest    0.6193  0.5997     0.7610    0.6708
          GBM    0.6274  0.6039     0.7703    0.6770
     Adaboost    0.6279  0.6032     0.7715    0.6770
      XGBoost    0.5996  0.5921     0.7371    0.6567
```

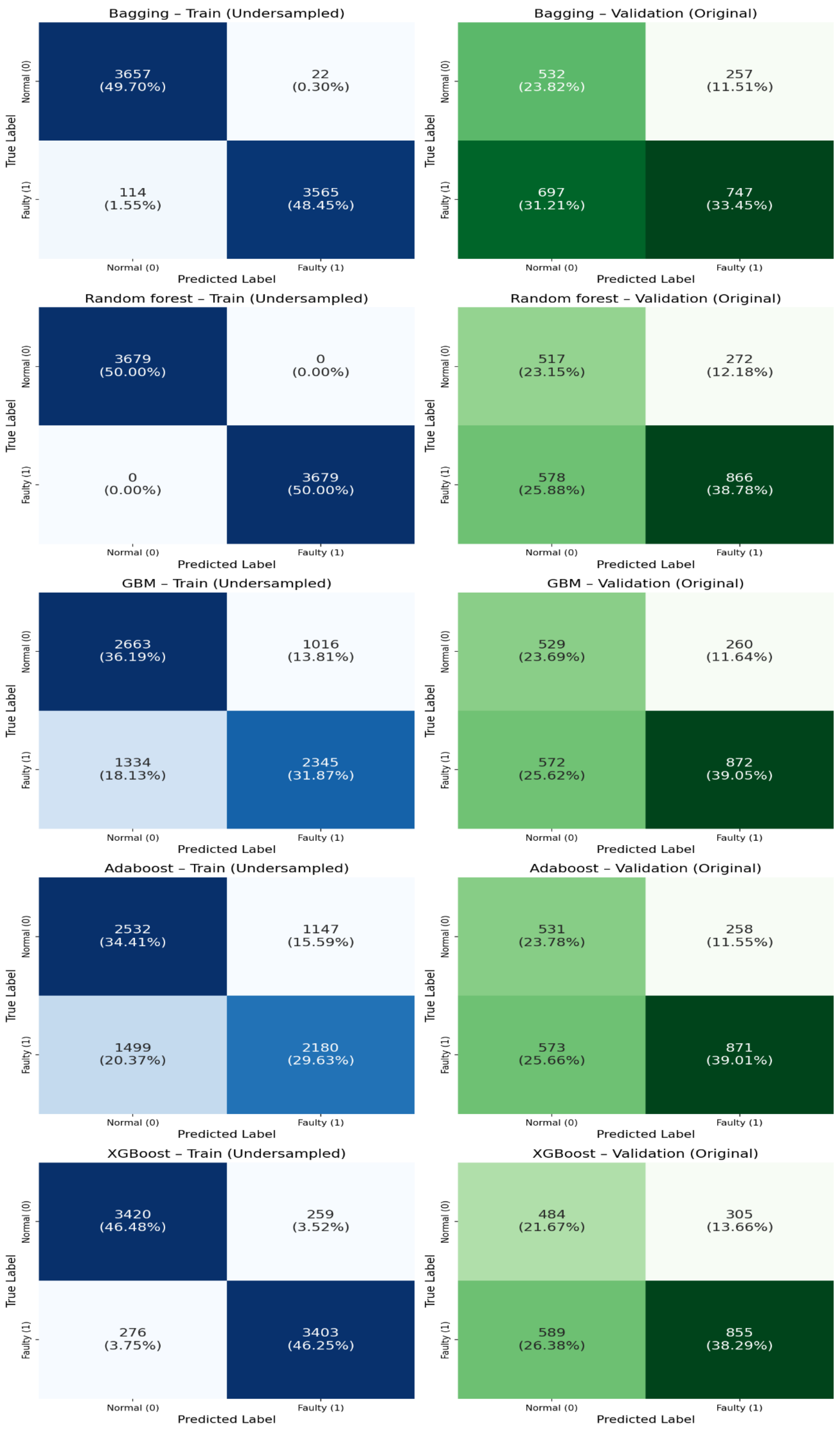


Fig 55: confusion matrix for 5 models built on Undersample Data for Training s Validating Data, respectively