Final Term Project

Jyothi Sevakula

Virginia Polytechnic Institute and State University

CS 5805 Machine Learning 1

Professor Dr. Reza Jafari

December 8th, 2024

Table of Contents

Table of figures

Abstract

Flight delays pose significant challenges in the airline industry, impacting both operational efficiency and passenger satisfaction. Delays can lead to cascading operational disruptions, increased costs for airlines, and frustration among passengers, highlighting the critical need for accurate and timely predictions. This project addresses this pressing issue by focusing on predicting flight delays using a comprehensive airlines dataset and a wide array of machine learning techniques. The ability to anticipate delays allows airlines to proactively manage schedules, allocate resources more effectively, and improve the overall passenger experience by providing timely updates and solutions.

The analysis employs regression, classification, clustering, and association rule mining to uncover patterns and insights that drive delay predictions. Through rigorous feature engineering, preprocessing, and dimensionality reduction techniques such as Random Forest, PCA, and SVD, the dataset is optimized for effective modeling. By identifying the most significant factors contributing to delays, this study provides actionable insights that can assist airlines in mitigating delays and enhancing operational reliability.

This project underscores the broader importance of leveraging predictive analytics within the aviation industry, demonstrating its potential to streamline operations, reduce costs, and boost customer satisfaction. The findings serve as a foundation for data-driven decision-making, offering a roadmap for implementing advanced analytics to address complex challenges in aviation.

Introduction

The project focuses on leveraging diverse machine learning methodologies to predict flight delays using a reliable airlines dataset based on scheduled flight details. The process begins with comprehensive preprocessing and feature engineering to ensure the dataset is prepared for model development.

Various dimensionality reduction techniques, including PCA, Random Forest, SVD, and VIF, are applied to streamline the feature space. The statistical significance of features is assessed through T-test and F-test analyses. Linear regression and backward stepwise regression are utilized to predict the numerical feature ‘Length’ and refine the model.

For classification, a range of models—such as decision trees, logistic regression, KNN, SVM, Naive Bayes, and neural networks—are evaluated. Grid search and stratified k-fold cross-validation are performed to identify the best hyperparameters, with model selection guided by a detailed analysis of evaluation metrics. Additionally, the project employs KMeans clustering, an unsupervised learning approach, to identify optimal clusters. Techniques like silhouette analysis and within-cluster sum of squares (WCSS) are used to determine the most appropriate number of clusters.

The project concludes with the application of the Apriori algorithm to uncover meaningful association rules within the dataset. This in-depth analysis provides valuable insights into the intricate relationships between features, enhancing the understanding of factors influencing flight delays.

Description of Dataset

The Airlines dataset comprises 539,383 instances and 9 distinct features, aimed at predicting whether a flight will be delayed based on its scheduled departure details. The dataset includes the following features: id, Airline, Flight, AirportFrom, AirportTo, DayOfWeek, Time, Length, and Delay.

Each record is uniquely identified by the id attribute and contains comprehensive flight information. The Airline attribute specifies the carrier operating the flight, while Flight represents the type of aircraft used. The attributes AirportFrom and AirportTo indicate the departure and arrival airports, respectively.

The DayOfWeek feature expressed numerically from 1 to 7, denotes the day the flight is scheduled. Time records the departure time in minutes past midnight, providing a precise temporal reference. The Length attribute reflects the flight duration in minutes, offering insights into its time span. Finally, the Delay attribute indicates whether the flight experienced a delay, serving as a key factor impacting airline operations and passenger satisfaction.

The dataset includes both categorical features (Airline, Flight, AirportFrom, AirportTo, DayOfWeek, Delay) and numerical features (Time, Length), meeting the project criteria of having more than 50,000 observations, at least two numerical features, and at least two categorical features with multiple categories.

Phase I: Feature Engineering and EDA

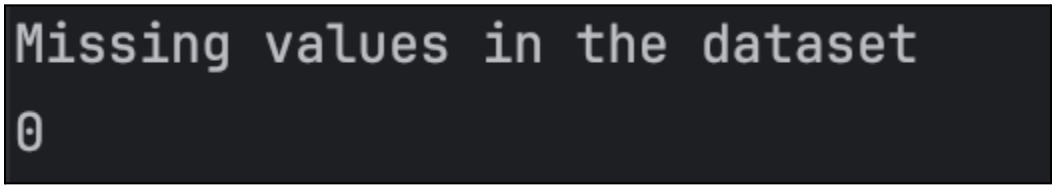
Data Preprocessing:

Data Cleaning:

There are no missing values in the dataset. There are also no nan values. The dataset is inherently clean and the information of the scheduled flights is consistent.

Figure 1

Missing values in the dataset

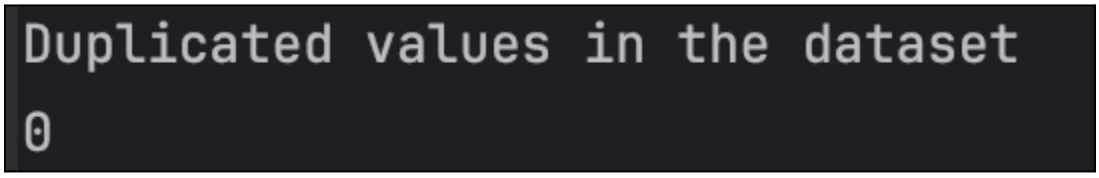


Data Duplicates:

There are no data duplicates in the dataset.

Figure 2

Duplicate values in the dataset



Down Sampling:

There are 18 categories in Airline, 293 categories in AirportFrom and 293 categories in AirportTo, 7 categories in DayOfWeek. Hence, I preferred to predict delays related to top 5 source airports which are from the AirportFrom feature and hence dropped the AirportTo feature. Also, considered the top 5 airlines as the remaining airlines have significantly low observations.

Figure 3

Unique values in each feature

```
Number of unique values in each feature of whole dataset
id             539383
Airline            18
Flight           6585
AirportFrom       293
AirportTo         293
DayOfWeek           7
Time             1131
Length            426
```

Now, the number of observations is 37,570 with 7 features. Also, dropped id columns as it has no significance in prediction.

Figure 4

Shape of the data after downsampling

```
After down sampling, the shape of the dataset
(37570, 7)
```

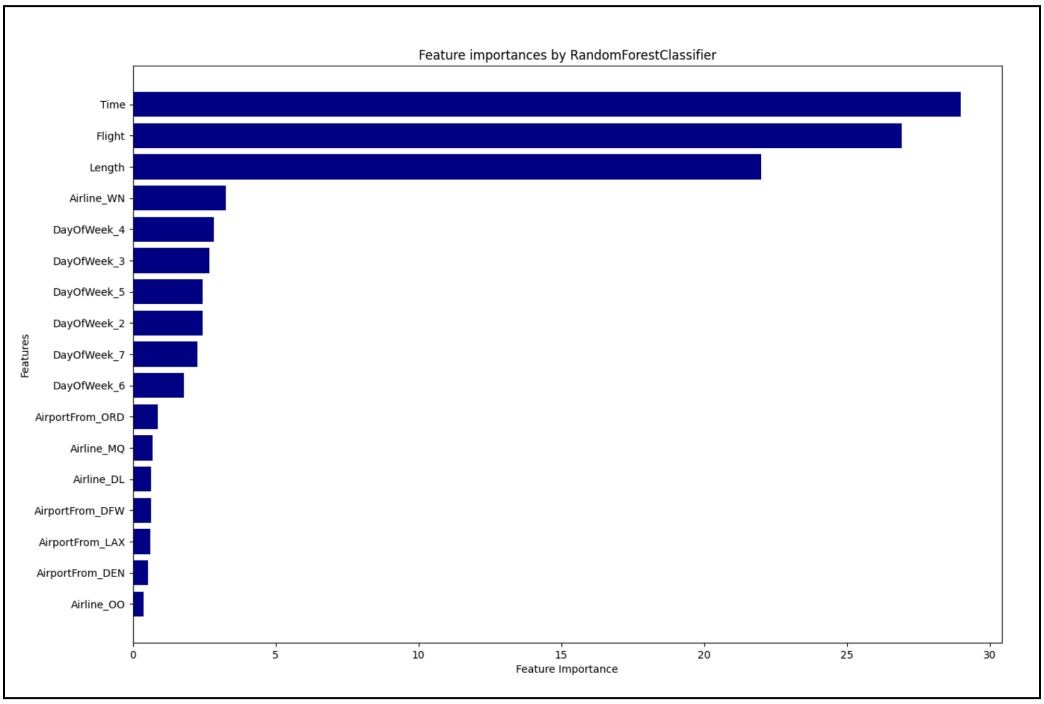
Dimensionality Reduction or Feature Selection:

Random Forest analysis:

Random Forest analysis plays a critical role in dimensionality reduction by identifying the most influential features that impact the model's predictions. This technique evaluates the relevance and contribution of each feature to the target variable using the Random Forest algorithm's built-in "feature_importances_" attribute. By quantifying the importance of each feature, this analysis highlights the key variables that significantly affect the model's decision-making process. Prioritizing these essential features helps streamline the feature space, resulting in a more efficient model while maintaining predictive accuracy. Retaining only the most impactful features is crucial for enhancing model performance and reducing computational complexity.

Figure 5

Plot of Decreasing order of feature importances from random forest



From above figure 5, we can see the importance of features in decreasing order. We can drop features by selecting a cumulative feature importance threshold of 95% and hence dropped features Airline_MQ, AirportFrom_DFW, AirportFrom_LAX, Airline_DL, AirportFrom_DEN, Airline_OO. Hence, the total dimensions are reduced to 11 from 17.

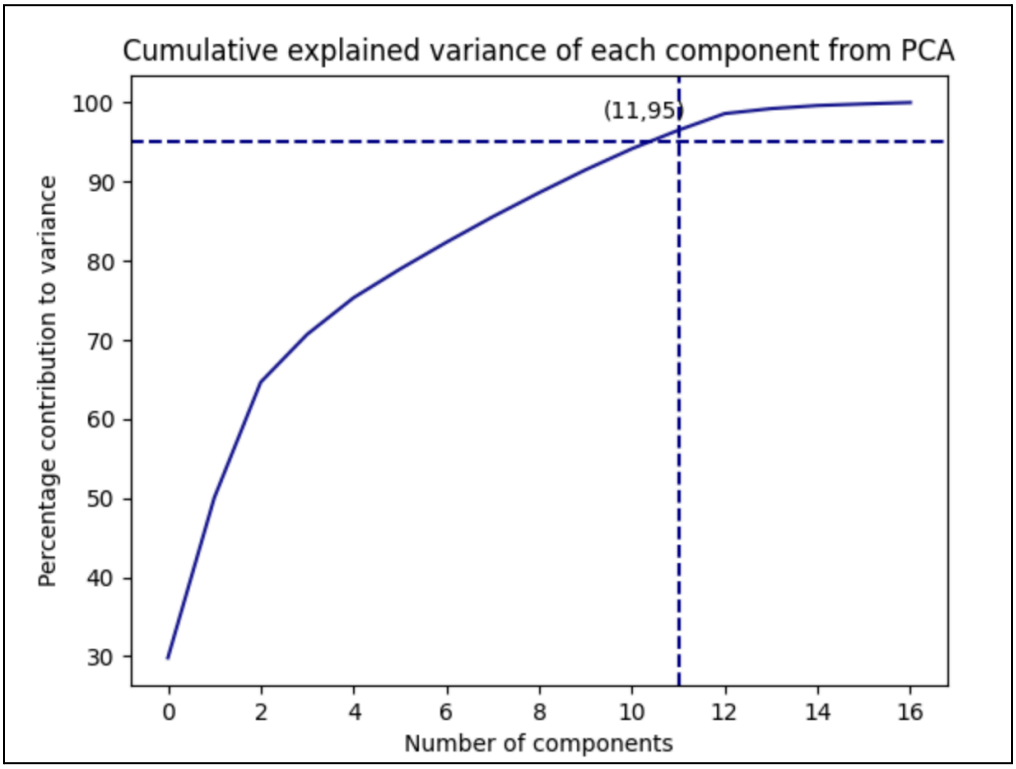
Principal Component Analysis:

Principal Component Analysis (PCA) is a widely used unsupervised learning technique for reducing the dimensionality of large datasets. It enhances interpretability while minimizing information loss, making it a fundamental tool in dimensionality reduction for data analysis. PCA works by transforming high-dimensional data into a lower-dimensional representation, called principal components. These components retain the most significant information from the original dataset, helping to condense the feature space while preserving key patterns and minimizing redundancy. Unlike methods that identify specific important features, PCA does not indicate which features are important but determines how many are essential to explain the majority of the variance. This dimensionality reduction facilitates easier visualization, and computational efficiency, and often improves the performance of machine learning models.

```
Cumulative explained variance of each component from PCA
[ 29.81  50.05  64.65  70.69  75.32  78.94  82.33  85.56  88.6   91.49
  94.16  96.5   98.6   99.23  99.62  99.82 100.  ]
```

Figure 6

Cumulative explained variance plot of PCA



When we choose a threshold of 95% ofthe total variance, we see that we are left with 11 components from PCA and these features are enough to explain 95% of the total variance of the data.

Singular Value Decomposition Analysis:

Singular Value Decomposition (SVD) is a robust dimensionality reduction technique in data analysis. It breaks down a matrix into three component matrices, uncovering patterns and structures within the data. By isolating the most significant components, SVD transforms high-dimensional data into a reduced set of singular vectors. This transformation creates a more compact representation of the original data while retaining critical information. SVD is instrumental in tasks like noise reduction, feature extraction, and improving computational efficiency in machine learning workflows.

Figure 7

Cumulative explained variance ratio plot from SVD

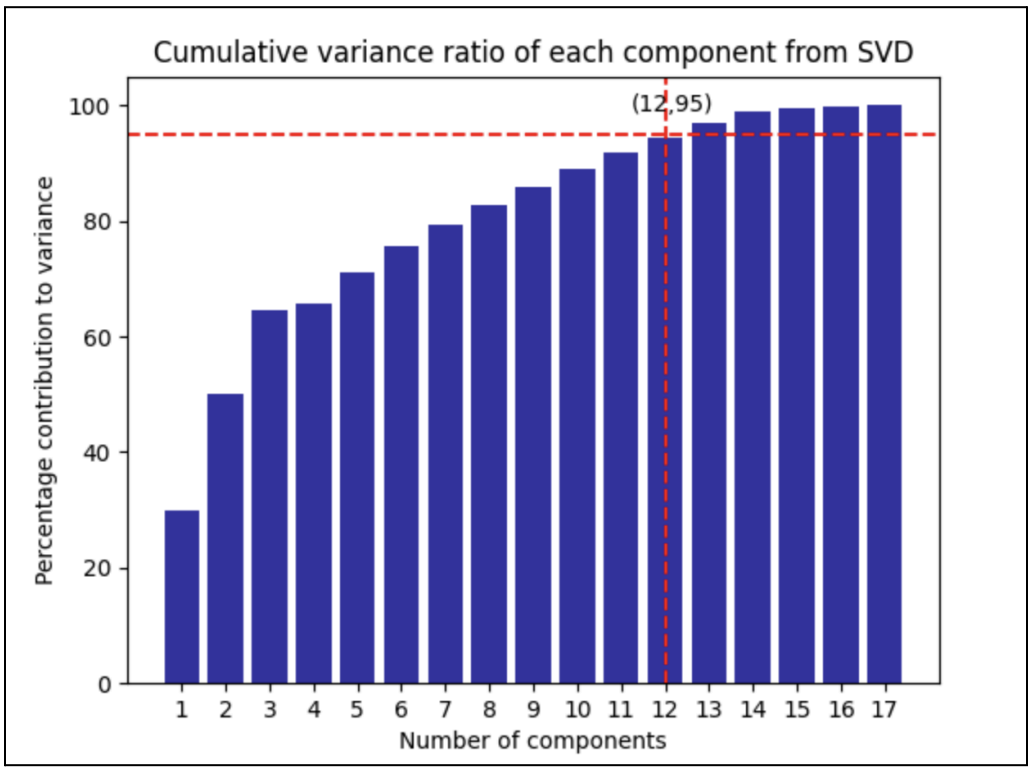


Figure 8

Singular values of the dataset

```
Singular values of features [229.87 189.3  160.73 129.39 100.8   90.46  80.09  77.45  75.46  73.35
  71.41  68.69  64.29  60.22  33.2   24.01  18.82]
```

Using a 95% threshold for total explained variance, the dimensions can be reduced from 17 to 12 without significant loss of information. This reduction simplifies the dataset, making it more manageable for further analysis..

Variance Inflation Factor:

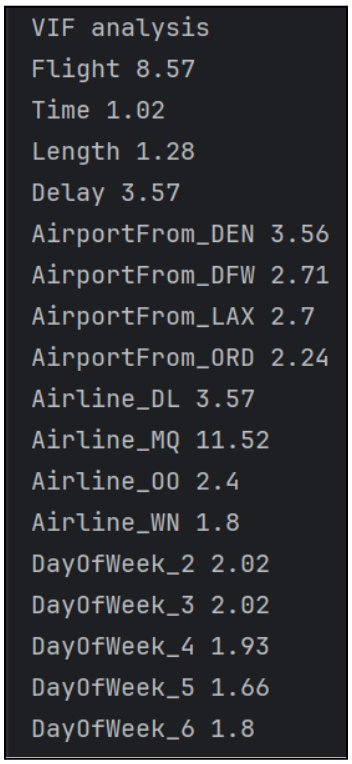
Variance Inflation Factor (VIF) is a tool used to identify multicollinearity among predictors in regression analysis. It measures how much the variance of a regression coefficient is increased due to multicollinearity among the predictors.

VIF values that are particularly high (typically above 5 or 10) often indicate strong multicollinearity, suggesting that certain predictors are either redundant or highly correlated. Removing these highly correlated features can help reduce multicollinearity and may serve as a form of dimensionality reduction in regression modeling.

Based on the VIF values observed, the variable 'Airline_MQ' shows significant multicollinearity with other predictors, which could affect the stability and reliability of the coefficient estimates in the regression model.

Figure 9

VIF values of each feature



Collinearity analysis:

we chose to reduce the features from the random forest as it reduces the most number of features and hence, we reduce collinearity also as these are eliminated in the random forest as well.

Observations:

The features eliminated from random forest analysis are Airline_MQ, AirportFrom_DFW, AirportFrom_LAX, Airline_DL, AirportFrom_DEN, and Airline_OO and leaves us with 11 dimensions which is the least of all the dimensionality reduction techniques. Hence, I chose to follow random forest feature importances.

Discretization and Binarization:

One-hot encoding was applied to all categorical columns, with one column from each category dropped to avoid the dummy variable trap. The resulting data frame after one-hot encoding is shown below.

Figure 10

One hot-encoded dataset

One hot encoded dataset

	Flight	Time	Length	Delay	AirportFrom_DEN	AirportFrom_DFW	AirportFrom_LAX	AirportFrom_ORD	Airline_DL	Airline_MQ	Airline_OO	Airline_WN
2	2400	20	165	1	0	0	1	0	0	0	0	0
13	2408	55	170	0	0	0	1	0	0	0	0	0
118	1297	345	160	0	0	1	0	0	0	0	0	0
228	2301	360	150	1	0	0	0	1	0	0	0	0
232	548	360	240	0	0	0	1	0	0	0	0	0

DayOfWeek_2	DayOfWeek_3	DayOfWeek_4	DayOfWeek_5	DayOfWeek_6	DayOfWeek_7
0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0

Variable transformation:

The dataframe was standardized prior to any analysis, model training, or dimensionality reduction techniques. Numerical columns were scaled, and the standardized data is shown below.

Figure 8

Standardized dataset

Standardized dataset

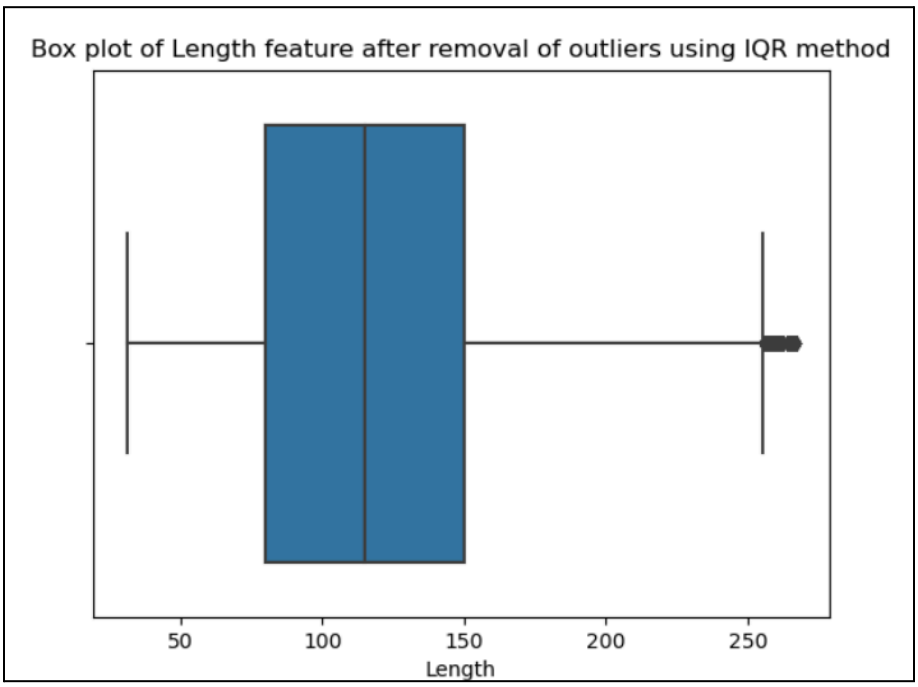
	Flight	Time	Length
2	-0.242986	-3.006184	0.860139
13	-0.239076	-2.879481	0.957252
118	-0.781981	-1.829653	0.763026
228	-0.291363	-1.775351	0.568800
232	-1.147990	-1.775351	2.316836

Outlier analysis and removal:

The numerical column ‘Length’ was examined for outliers, which were removed using the IQR method. This approach involves calculating the first quartile (Q1) and third quartile (Q3) of the dataset. The interquartile range (IQR) is then determined, and the lower and upper bounds are defined as Q1 minus 1.5 times the IQR and Q3 plus 1.5 times the IQR, respectively. Any values outside these bounds are classified as outliers and removed.

Figure 11

Box plot of Length feature after removal of outliers using IQR method



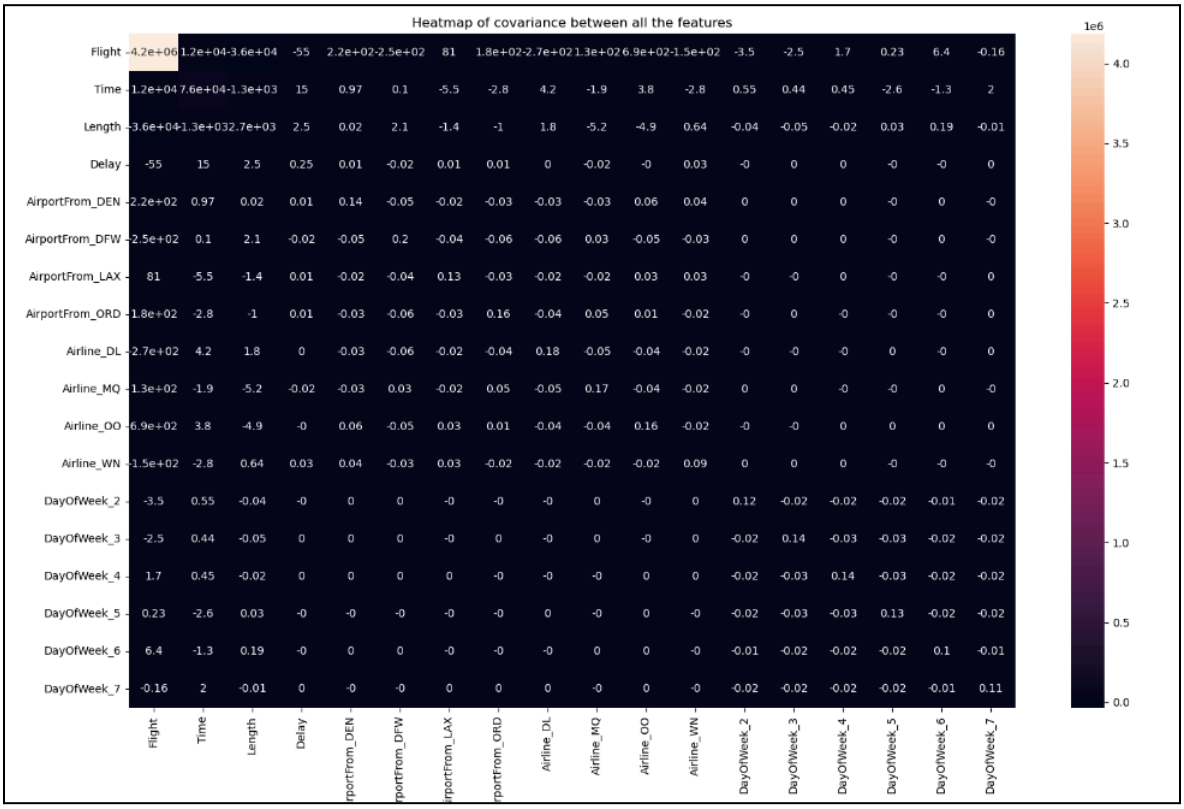
This box plot shows the Length after outlier removal. The outliers currently present are with respect to the updated column where the IQR, Q1 and Q3 changed. Surprisingly, this also removed outliers in the 'Time' column.

Covariance Matrix:

Covariance is a statistical measure that indicates the extent to which two variables change together. It helps assess the direction of the linear relationship between two variables. The heatmap below shows the sample covariance matrix of the dataset across all features. The plot reveals that most values are quite small, likely because the majority of the features are categorical, resulting in minimal relationships between them.

Figure 12

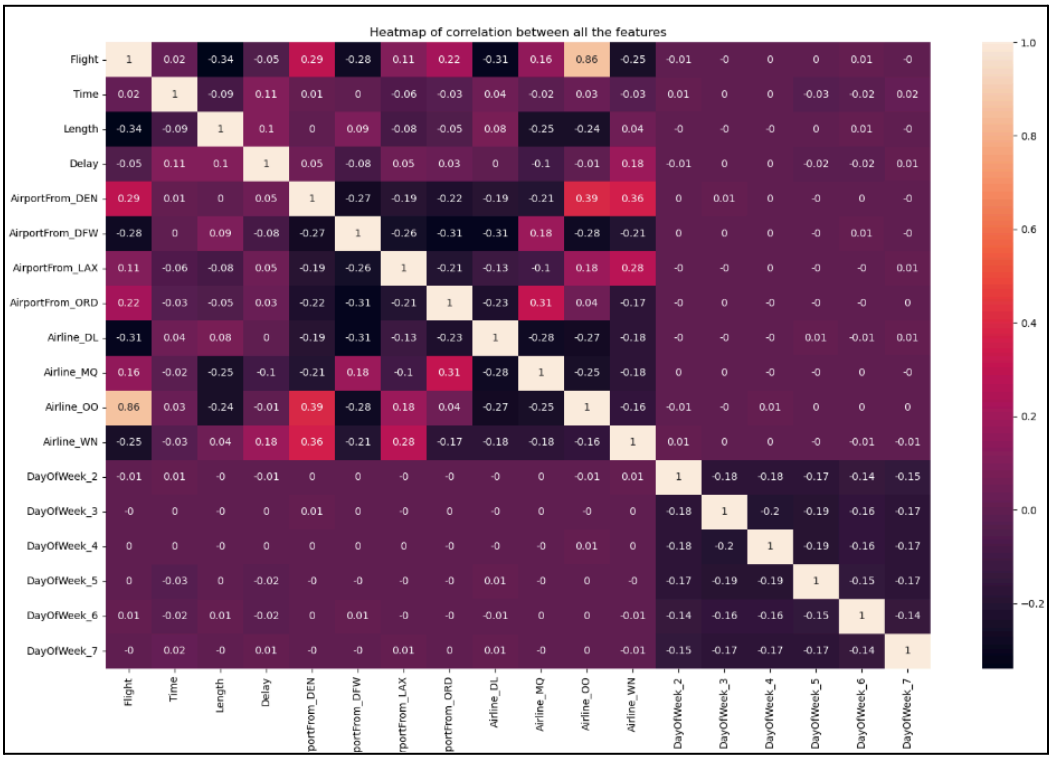
The covariance matrix of all features



Pearson correlation matrix:

Figure 13

Pearson correlation coefficient matrix



The Pearson correlation matrix indicates that most values are close to 0, with a few features having correlations between 0.2 and 0.4. This suggests minimal correlation between features, primarily because most of them are categorical.

Balanced or Imbalanced data:

Values in each class of the target variable are almost equal and hence, data is balanced with no bias problem in any further analysis.

Figure 14

Count of each class of target 'Delay'

```
The value counts of each class of target feature
Delay
0    19806
1    17764
```

Phase II: Regression Analysis

T-test analysis:

The t-test is a statistical technique used to determine whether there is a significant difference between the means of two groups. It evaluates whether the observed differences are due to actual differences in the population or merely random chance. Among the various types of t-tests, the two-sample t-test is the most commonly used and compares the means of two independent groups to determine if they are significantly different. This is achieved by analyzing the group means, variances, and calculating a t-statistic, which is then compared to a critical value from the t-distribution at a chosen significance level (usually 0.05). If the t-value exceeds this critical value, it indicates that the observed difference is unlikely to result from random variation, signifying a statistically significant difference between the groups.

In this analysis, features such as Time, Delay, AirportFrom_DEN, AirportFrom_DFW, AirportFrom_LAX, AirportFrom_ORD, Airline_MQ, Airline_OO, and Airline_WN exhibit very low p-values (close to 0), highlighting their strong significance in predicting flight delays. Conversely, features like DayOfWeek_2, DayOfWeek_3, DayOfWeek_4, DayOfWeek_5, DayOfWeek_6, and DayOfWeek_7 have higher p-values (above 0.05), indicating they may not significantly influence flight delay predictions based on this analysis.

Figure 15

T-statistic and p-values of features

t-test analysis

Feature	t-statistic	p-value
const	2.22	0.03
Flight	-1.52	0.13
Time	-16.24	0.0
Delay	15.11	0.0
AirportFrom_DEN	27.45	0.0
AirportFrom_DFW	12.54	0.0
AirportFrom_LAX	15.57	0.0
AirportFrom_ORD	18.46	0.0
Airline_DL	-3.52	0.0
Airline_MQ	-45.77	0.0
Airline_OO	-27.38	0.0
Airline_WN	-28.71	0.0
DayOfWeek_2	0.45	0.65
DayOfWeek_3	-0.17	0.86
DayOfWeek_4	0.25	0.8
DayOfWeek_5	0.43	0.67
DayOfWeek_6	1.48	0.14
DayOfWeek_7	0.75	0.45

F-test analysis:

The F-test is a statistical technique used to compare the variability of two or more groups to determine whether their variances differ significantly. It assesses whether the observed variances within these groups are equal or if the differences exceed what can be attributed to random variation. In the context of Analysis of Variance (ANOVA), the F-test is frequently used to analyze the variances of multiple groups simultaneously. The F-statistic is calculated as the ratio of variance between groups to variance within groups, helping to identify if the variability between group means is significantly larger than within-group variability. By comparing the F-value to a critical value from the F-distribution at a chosen significance level (typically 0.05), it can be determined whether the variance differences are statistically significant.

In this analysis, features with high F-test scores, such as Flight, Time, Airline_DL, Airline_MQ, and Airline_OO, demonstrate greater importance in explaining the variability of the target variable. Conversely, features with low F-test scores, such as DayOfWeek_2, DayOfWeek_3, DayOfWeek_4, DayOfWeek_5, and DayOfWeek_7, show lesser significance in contributing to the variability. Additionally, p-values close to 0.0 provide strong evidence against the null hypothesis, indicating that the overall set of predictors plays a significant role in explaining the target variable.

Figure 16

F-test and p-values of each feature

Feature	F-test score	p-value
Flight	89.3	0.0
Time	14.91	0.0
Length	4.03	0.0
AirportFrom_DEN	37.04	0.0
AirportFrom_DFW	48.37	0.0
AirportFrom_LAX	65.57	0.0
AirportFrom_ORD	24.22	0.0
Airline_DL	112.05	0.0
Airline_MQ	65.06	0.0
Airline_OO	122.7	0.0
Airline_WN	23.58	0.0
DayOfWeek_2	0.53	1.0
DayOfWeek_3	0.67	1.0
DayOfWeek_4	0.58	1.0
DayOfWeek_5	0.73	1.0
DayOfWeek_6	1.25	0.01
DayOfWeek_7	0.97	0.62

Linear regression model and prediction of dependent variable:

Linear regression is a fundamental statistical method used to model the relationship between a dependent variable (y) and one or more independent variables (x). Its main goal is to analyze how variations in the independent variables influence the dependent variable. The model operates under the assumption of a linear relationship between the independent and dependent variables.

Figure 17

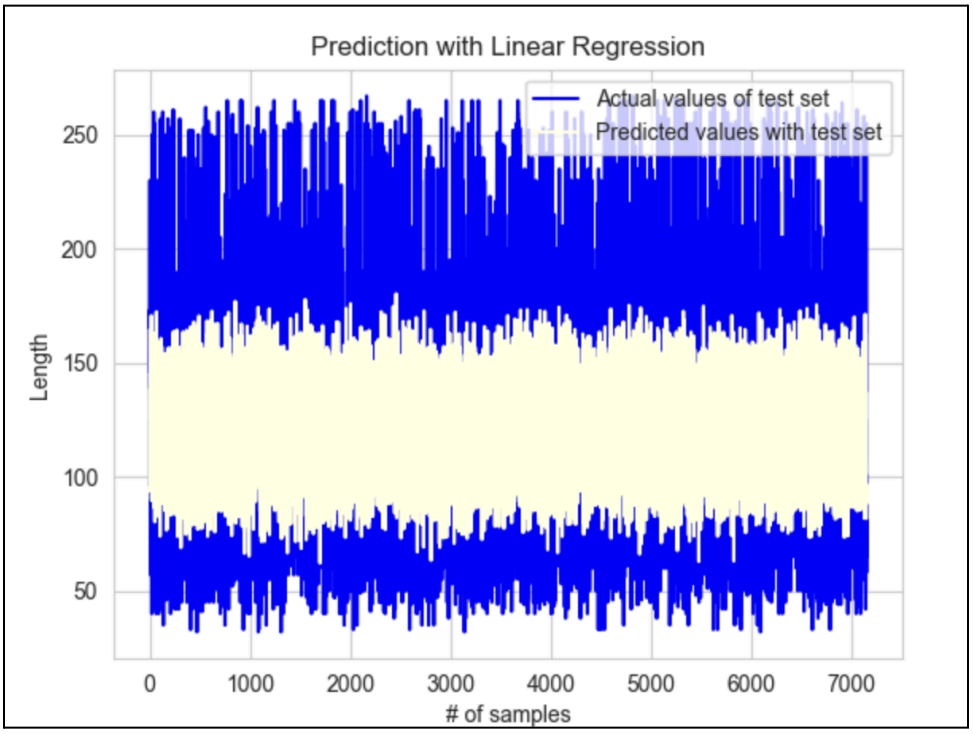
Coefficients and R2, Adj. R2, AIC, BIC, MSE of the Linear Regression Model

```
The coefficients of the Linear Regression model
Feature Coefficient
const 0.0
Flight -0.02
Time -0.09
Delay 0.16
AirportFrom_DEN 0.8
AirportFrom_DFW 0.35
AirportFrom_LAX 0.43
AirportFrom_ORD 0.51
Airline_DL -0.09
Airline_MQ -1.04
Airline_OO -1.22
Airline_WN -0.74
DayOfWeek_2 0.01
DayOfWeek_3 -0.0
DayOfWeek_4 0.0
DayOfWeek_5 0.01
DayOfWeek_6 0.03
DayOfWeek_7 0.02
Mean Squared Error for training set (MSE): 2054.446
+------------------+-----------+---------------+-----------+-----------+--------------------+
|       Model      | R-squared | Adj. R-squared |    AIC    |    BIC    | Mean Squared Error |
+------------------+-----------+---------------+-----------+-----------+--------------------+
| Linear Regression |    0.23   |      0.22      | 218035.43 | 218175.86 |      2054.45       |
+------------------+-----------+---------------+-----------+-----------+--------------------+
```

Linear regression was performed on the original feature set after standardizing the numerical features and the target variable, ‘Length’. The intercept value of 0.0 indicates that the predicted value of the target is zero when all other predictors are zero. The model explains a moderate portion of the variability in the target variable ‘Length’ which is around 23%.

Figure 18

Predicted and Test values of 'Length' using Linear Regression



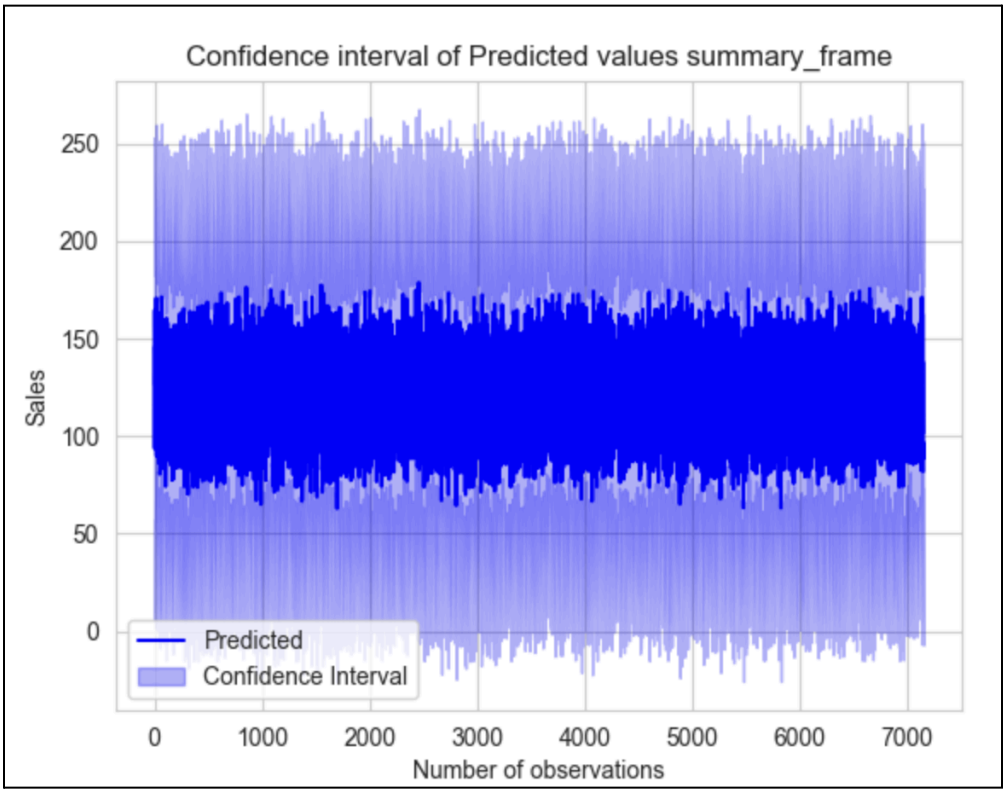
The training and predicted values can be seen in the above graph and the model has bias between predicted and training values.

Confidence interval analysis:

The below plot shows the confidence interval of the predicted values which has huge variance from the actual predicted values. This indicates that there is uncertainty in the estimated coefficients and this model is not reliable in predicting the 'Length'. This uncertainty is expected as our dataset is a classification problem and this is an independent variable and helps in predicting Delay rather than being predicted.

Figure 19

Confidence Interval analysis



Stepwise regression and adjusted R-square analysis:

This method begins with a model that includes all predictors, gradually removing the least significant variables until no further improvement in the model's performance is achieved. Backward stepwise regression is chosen as some features exhibit higher p-values in the T-test. Features with the highest p-value, exceeding the threshold of 0.01, are eliminated step by step. This process reduces the number of features while retaining those sufficient to explain the variance in the target variable. At each step, the Adjusted R-squared from the OLS summary is monitored, and the process is halted if its value drops, indicating the need to retain the recently removed feature.

Features DayOfWeek_3, DayOfWeek_4, DayOfWeek_5, DayOfWeek_2, DayOfWeek_7, Flight, and DayOfWeek_6 eliminated.

```
eliminated features: ['DayofWeek_3', 'DayofWeek_4', 'DayofWeek_5', 'DayofWeek_2', 'DayOfWeek_7', 'Flight', 'DayOfWeek_6']
```

Figure 20

OLS summary after eliminating the 'DayOfWeek_3' feature

```
Eliminating the feature with highest p-value which is DayOfWeek_3
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.225
Model:                            OLS   Adj. R-squared:                  0.225
Method:                 Least Squares   F-statistic:                     519.3
Date:                Mon, 02 Dec 2024   Prob (F-statistic):               0.00
Time:                        11:50:43   Log-Likelihood:                -36911.
No. Observations:               28580   AIC:                         7.386e+04
Df Residuals:                   28563   BIC:                         7.400e+04
Df Model:                          16
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0715      0.031      2.286      0.022       0.010       0.133
Flight        -0.0248      0.016     -1.523      0.128      -0.057       0.007
Time          -0.0859      0.005    -16.242      0.000      -0.096      -0.076
Delay          0.1625      0.011     15.114      0.000       0.141       0.184
AirportFrom_DEN  0.8021    0.029     27.448      0.000       0.745       0.859
AirportFrom_DFW  0.3504    0.028     12.539      0.000       0.296       0.405
AirportFrom_LAX  0.4303    0.028     15.572      0.000       0.376       0.484
AirportFrom_ORD  0.5138    0.028     18.465      0.000       0.459       0.568
Airline_DL    -0.0948      0.027     -3.522      0.000      -0.147      -0.042
Airline_MQ    -1.0358      0.023    -45.766      0.000      -1.080      -0.991
Airline_OO    -1.2170      0.044    -27.379      0.000      -1.304      -1.130
Airline_WN    -0.7431      0.026    -28.709      0.000      -0.794      -0.692
DayOfWeek_2    0.0110      0.017      0.641      0.521      -0.023       0.045
DayOfWeek_4    0.0066      0.016      0.416      0.678      -0.025       0.038
DayOfWeek_5    0.0102      0.016      0.625      0.532      -0.022       0.042
DayOfWeek_6    0.0332      0.018      1.823      0.068      -0.002       0.069
DayOfWeek_7    0.0171      0.017      0.987      0.324      -0.017       0.051
```

Figure 21

OLS summary after eliminating 'DayOfWeek_4' feature

Eliminating DayOfWeek_4

OLS Regression Results

Dep. Variable:	y	R-squared:	0.225
Model:	OLS	Adj. R-squared:	0.225
Method:	Least Squares	F-statistic:	553.9
Date:	Mon, 02 Dec 2024	Prob (F-statistic):	0.00
Time:	11:50:43	Log-Likelihood:	-36911.
No. Observations:	28580	AIC:	7.385e+04
Df Residuals:	28564	BIC:	7.399e+04
Df Model:	15		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0740	0.031	2.409	0.016	0.014	0.134
Flight	-0.0248	0.016	-1.521	0.128	-0.057	0.007
Time	-0.0859	0.005	-16.241	0.000	-0.096	-0.076
Delay	0.1625	0.011	15.113	0.000	0.141	0.184
AirportFrom_DEN	0.8019	0.029	27.445	0.000	0.745	0.859
AirportFrom_DFW	0.3503	0.028	12.536	0.000	0.296	0.405
AirportFrom_LAX	0.4302	0.028	15.569	0.000	0.376	0.484
AirportFrom_ORD	0.5137	0.028	18.462	0.000	0.459	0.568
Airline_DL	-0.0948	0.027	-3.524	0.000	-0.148	-0.042
Airline_MQ	-1.0358	0.023	-45.767	0.000	-1.080	-0.991
Airline_OO	-1.2169	0.044	-27.378	0.000	-1.304	-1.130
Airline_WN	-0.7430	0.026	-28.707	0.000	-0.794	-0.692
DayOfWeek_2	0.0086	0.016	0.533	0.594	-0.023	0.040
DayOfWeek_5	0.0078	0.015	0.512	0.609	-0.022	0.038
DayOfWeek_6	0.0309	0.017	1.782	0.075	-0.003	0.065
DayOfWeek_7	0.0147	0.016	0.900	0.368	-0.017	0.047

Figure 22

OLS summary after eliminating 'DayOfWeek_5' feature

Eliminating DayOfWeek_5

OLS Regression Results

Dep. Variable:	y	R-squared:	0.225
Model:	OLS	Adj. R-squared:	0.225
Method:	Least Squares	F-statistic:	593.5
Date:	Mon, 02 Dec 2024	Prob (F-statistic):	0.00
Time:	11:50:43	Log-Likelihood:	-36911.
No. Observations:	28580	AIC:	7.385e+04
Df Residuals:	28565	BIC:	7.398e+04
Df Model:	14		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0760	0.030	2.497	0.013	0.016	0.136
Flight	-0.0247	0.016	-1.517	0.129	-0.057	0.007
Time	-0.0859	0.005	-16.255	0.000	-0.096	-0.076
Delay	0.1624	0.011	15.106	0.000	0.141	0.183
AirportFrom_DEN	0.8018	0.029	27.442	0.000	0.745	0.859
AirportFrom_DFW	0.3503	0.028	12.536	0.000	0.296	0.405
AirportFrom_LAX	0.4301	0.028	15.567	0.000	0.376	0.484
AirportFrom_ORD	0.5137	0.028	18.461	0.000	0.459	0.568
Airline_DL	-0.0948	0.027	-3.523	0.000	-0.148	-0.042
Airline_MQ	-1.0359	0.023	-45.770	0.000	-1.080	-0.992
Airline_OO	-1.2170	0.044	-27.381	0.000	-1.304	-1.130
Airline_WN	-0.7429	0.026	-28.705	0.000	-0.794	-0.692
DayOfWeek_2	0.0067	0.016	0.425	0.671	-0.024	0.037
DayOfWeek_6	0.0289	0.017	1.712	0.087	-0.004	0.062
DayOfWeek_7	0.0128	0.016	0.804	0.422	-0.018	0.044

Figure 23

OLS summary after eliminating 'DayOfWeek_2' feature

Eliminating DayOfWeek_2

OLS Regression Results

Dep. Variable:	y	R-squared:	0.225
Model:	OLS	Adj. R-squared:	0.225
Method:	Least Squares	F-statistic:	639.2
Date:	Mon, 02 Dec 2024	Prob (F-statistic):	0.00
Time:	11:50:43	Log-Likelihood:	-36911.
No. Observations:	28580	AIC:	7.385e+04
Df Residuals:	28566	BIC:	7.397e+04
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0773	0.030	2.550	0.011	0.018	0.137
Flight	-0.0247	0.016	-1.515	0.130	-0.057	0.007
Time	-0.0859	0.005	-16.251	0.000	-0.096	-0.076
Delay	0.1624	0.011	15.102	0.000	0.141	0.183
AirportFrom_DEN	0.8018	0.029	27.442	0.000	0.745	0.859
AirportFrom_DFW	0.3503	0.028	12.536	0.000	0.296	0.405
AirportFrom_LAX	0.4301	0.028	15.566	0.000	0.376	0.484
AirportFrom_ORD	0.5137	0.028	18.462	0.000	0.459	0.568
Airline_DL	-0.0948	0.027	-3.525	0.000	-0.148	-0.042
Airline_MQ	-1.0359	0.023	-45.773	0.000	-1.080	-0.992
Airline_OO	-1.2171	0.044	-27.385	0.000	-1.304	-1.130
Airline_WN	-0.7429	0.026	-28.704	0.000	-0.794	-0.692
DayOfWeek_6	0.0277	0.017	1.665	0.096	-0.005	0.060
DayOfWeek_7	0.0116	0.016	0.741	0.459	-0.019	0.042

Figure 24

OLS summary after eliminating 'DayOfWeek_7' feature

```
Eliminating DayOfWeek_7
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.225
Model:                            OLS   Adj. R-squared:                  0.225
Method:                 Least Squares   F-statistic:                     692.4
Date:                Mon, 02 Dec 2024   Prob (F-statistic):               0.00
Time:                        11:50:43   Log-Likelihood:                -36911.
No. Observations:               28580   AIC:                         7.385e+04
Df Residuals:                   28567   BIC:                         7.396e+04
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const              0.0789      0.030      2.610      0.009       0.020       0.138
Flight            -0.0247      0.016     -1.517      0.129      -0.057       0.007
Time              -0.0858      0.005    -16.243      0.000      -0.096      -0.075
Delay              0.1624      0.011     15.108      0.000       0.141       0.183
AirportFrom_DEN    0.8018      0.029     27.445      0.000       0.745       0.859
AirportFrom_DFW    0.3503      0.028     12.538      0.000       0.296       0.405
AirportFrom_LAX    0.4302      0.028     15.571      0.000       0.376       0.484
AirportFrom_ORD    0.5138      0.028     18.466      0.000       0.459       0.568
Airline_DL        -0.0948      0.027     -3.524      0.000      -0.148      -0.042
Airline_MQ        -1.0360      0.023    -45.776      0.000      -1.080      -0.992
Airline_OO        -1.2171      0.044    -27.385      0.000      -1.304      -1.130
Airline_WN        -0.7431      0.026    -28.712      0.000      -0.794      -0.692
DayOfWeek_6        0.0260      0.017      1.578      0.115      -0.006       0.058
==============================================================================
```

Figure 25

OLS summary after eliminating 'Flight' feature

Eliminating Flight

OLS Regression Results

Dep. Variable:	y	R-squared:	0.225
Model:	OLS	Adj. R-squared:	0.225
Method:	Least Squares	F-statistic:	755.1
Date:	Mon, 02 Dec 2024	Prob (F-statistic):	0.00
Time:	11:50:43	Log-Likelihood:	-36912.
No. Observations:	28580	AIC:	7.385e+04
Df Residuals:	28568	BIC:	7.395e+04
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0964	0.028	3.449	0.001	0.042	0.151
Time	-0.0857	0.005	-16.215	0.000	-0.096	-0.075
Delay	0.1628	0.011	15.144	0.000	0.142	0.184
AirportFrom_DEN	0.8004	0.029	27.410	0.000	0.743	0.858
AirportFrom_DFW	0.3529	0.028	12.651	0.000	0.298	0.408
AirportFrom_LAX	0.4303	0.028	15.572	0.000	0.376	0.484
AirportFrom_ORD	0.5103	0.028	18.404	0.000	0.456	0.565
Airline_DL	-0.0984	0.027	-3.670	0.000	-0.151	-0.046
Airline_MQ	-1.0608	0.016	-67.740	0.000	-1.091	-1.030
Airline_OO	-1.2768	0.021	-61.630	0.000	-1.317	-1.236
Airline_WN	-0.7419	0.026	-28.680	0.000	-0.793	-0.691
DayOfWeek_6	0.0253	0.016	1.537	0.124	-0.007	0.058

Figure 26

OLS summary after eliminating DayOfWeek_6 feature

Eliminating DayOfWeek_6

OLS Regression Results

Dep. Variable:	y	R-squared:	0.225
Model:	OLS	Adj. R-squared:	0.225
Method:	Least Squares	F-statistic:	830.3
Date:	Mon, 02 Dec 2024	Prob (F-statistic):	0.00
Time:	11:50:43	Log-Likelihood:	-36914.
No. Observations:	28580	AIC:	7.385e+04
Df Residuals:	28569	BIC:	7.394e+04
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0995	0.028	3.571	0.000	0.045	0.154
Time	-0.0858	0.005	-16.243	0.000	-0.096	-0.075
Delay	0.1626	0.011	15.129	0.000	0.142	0.184
AirportFrom_DEN	0.8005	0.029	27.412	0.000	0.743	0.858
AirportFrom_DFW	0.3528	0.028	12.648	0.000	0.298	0.407
AirportFrom_LAX	0.4303	0.028	15.572	0.000	0.376	0.484
AirportFrom_ORD	0.5101	0.028	18.397	0.000	0.456	0.564
Airline_DL	-0.0987	0.027	-3.681	0.000	-0.151	-0.046
Airline_MQ	-1.0608	0.016	-67.745	0.000	-1.092	-1.030
Airline_OO	-1.2768	0.021	-61.633	0.000	-1.317	-1.236
Airline_WN	-0.7423	0.026	-28.697	0.000	-0.793	-0.692

Figure 27

OLS summary after Backward regression

OLS Regression Results

Dep. Variable:	y	R-squared:	0.225
Model:	OLS	Adj. R-squared:	0.225
Method:	Least Squares	F-statistic:	830.3
Date:	Mon, 02 Dec 2024	Prob (F-statistic):	0.00
Time:	11:50:43	Log-Likelihood:	-36914.
No. Observations:	28580	AIC:	7.385e+04
Df Residuals:	28569	BIC:	7.394e+04
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0995	0.028	3.571	0.000	0.045	0.154
Time	-0.0858	0.005	-16.243	0.000	-0.096	-0.075
Delay	0.1626	0.011	15.129	0.000	0.142	0.184
AirportFrom_DEN	0.8005	0.029	27.412	0.000	0.743	0.858
AirportFrom_DFW	0.3528	0.028	12.648	0.000	0.298	0.407
AirportFrom_LAX	0.4303	0.028	15.572	0.000	0.376	0.484
AirportFrom_ORD	0.5101	0.028	18.397	0.000	0.456	0.564
Airline_DL	-0.0987	0.027	-3.681	0.000	-0.151	-0.046
Airline_MQ	-1.0608	0.016	-67.745	0.000	-1.092	-1.030
Airline_OO	-1.2768	0.021	-61.633	0.000	-1.317	-1.236
Airline_WN	-0.7423	0.026	-28.697	0.000	-0.793	-0.692

Omnibus:	1539.012	Durbin-Watson:	2.009
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1798.650
Skew:	0.611	Prob(JB):	0.00
Kurtosis:	3.133	Cond. No.	14.6

We find that the features const, Time, Delay, AirportFrom_DEN, AirportFrom_DFW, AirportFrom_LAX, AirportFrom_ORD, Airline_DL, Airline_MQ, Airline_OO, Airline_WN are enough to perform linear regression and achieve Adj. R-squared as 23%.

Features DayOfWeek_3, DayOfWeek_4, DayOfWeek_5, DayOfWeek_2, DayOfWeek_7, Flight, and DayOfWeek_6 eliminated.

Figure 28

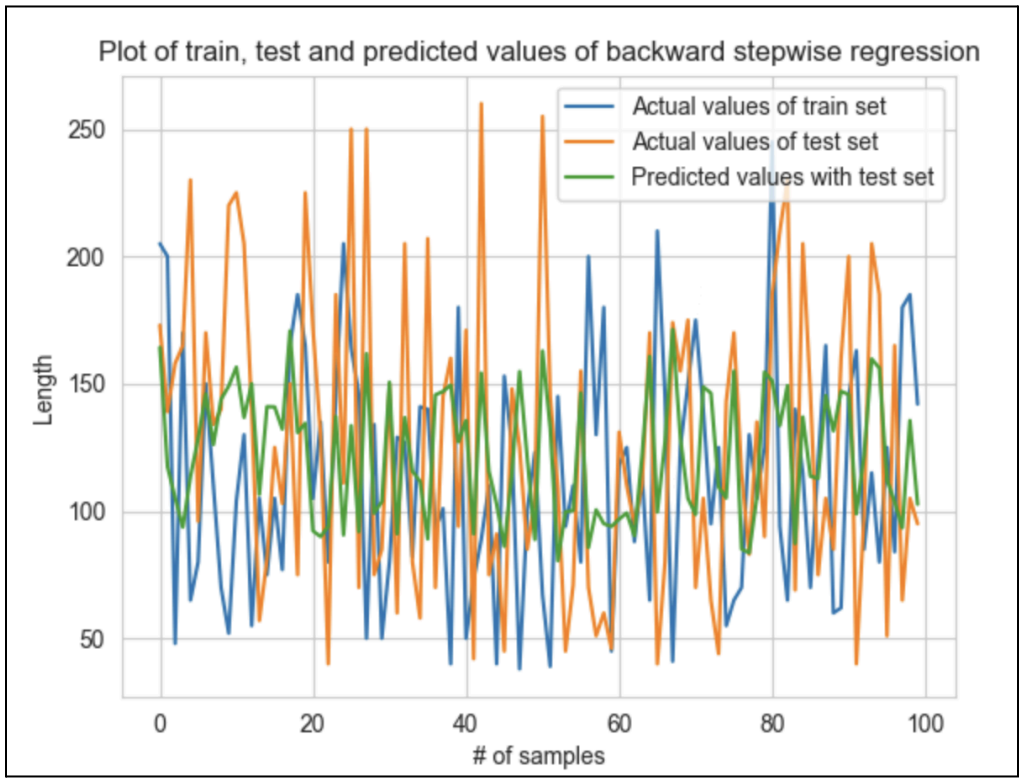
Condition number and Singular values after Backward stepwise regression

```
Condition number of the remaining features of the dataset after backward stepwise regression
14.65
Singular values of the remaining features of the dataset after backward stepwise regression
[212.6  169.2   96.36  84.95  81.07  76.89  66.15  61.43  49.75  24.56
  14.52]
```

The condition number and singular values suggest that there's no collinearity and hence, these are our final set of features that are sufficient.

Figure 29

Plot of train, test, and predicted values of Backward stepwise regression



The above plot shows the train, test, and predicted values after we find the final set of features from backward stepwise regression.

Comparing Linear regression and Backward stepwise regression

Figure 30

Comparison of Linear regression and Backward stepwise regression

Model	R-squared	Adj. R-squared	AIC	BIC	Mean Squared Error
Linear Regression	0.23	0.22	218035.43	218175.86	2054.45
Backward Stepwise Regression	0.23	0.22	73849.35	73940.22	2032.57

Both the analyses performed equally well in explaining the target variable's variability but, with backward regression, we significantly reduced AIC and BIC values and also reduced MSE. We are able to achieve the same results with reduced feature space in backward stepwise

Phase III: Classification Analysis

Decision Tree:

Pre-pruned decision tree:

Pruning techniques focus on balancing model complexity and predictive accuracy. To prevent the decision tree from overfitting by growing unchecked and perfectly fitting the training data, pre-pruning introduces constraints on the tree's growth or depth during its construction. This ensures the tree does not expand to its maximum allowable depth. Grid search and stratified k-fold cross-validation were performed to train the model and identify the best hyperparameters.

Figure 31

Best hyperparameters of Pre-pruned decision tree

```
Best Hyperparameters: {'criterion': 'entropy', 'max_depth': 9, 'max_features': 7, 'min_samples_leaf': 4, 'min_samples_split': 10, 'splitter': 'best'}
```

Figure 32

Various metrics of Pre-pruned decision tree

Classifier	Accuracy	Confusion Matrix	Precision	Sensitivity or Recall	Specificity	F-score	AUC
Pre pruned Decision Tree Classifier	0.62	[[3047 743] [1956 1400]]	0.65	0.42	0.8	0.51	0.67

Figure 33

Confusion matrix of Pre-pruned decision tree

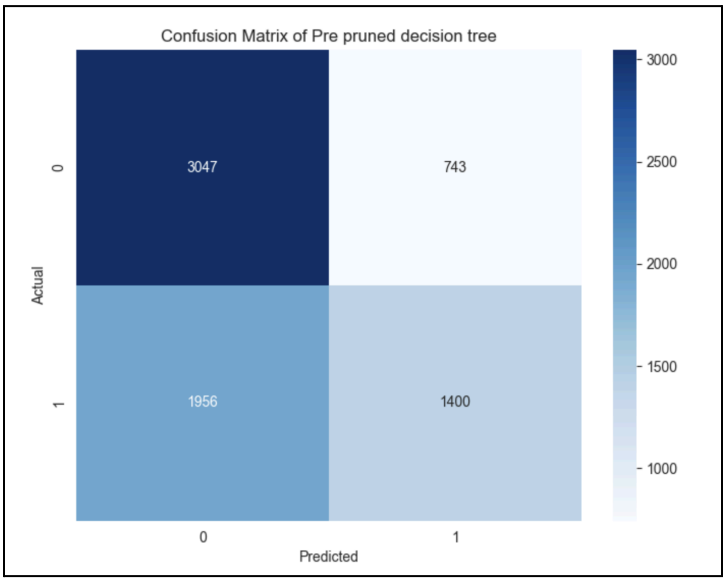
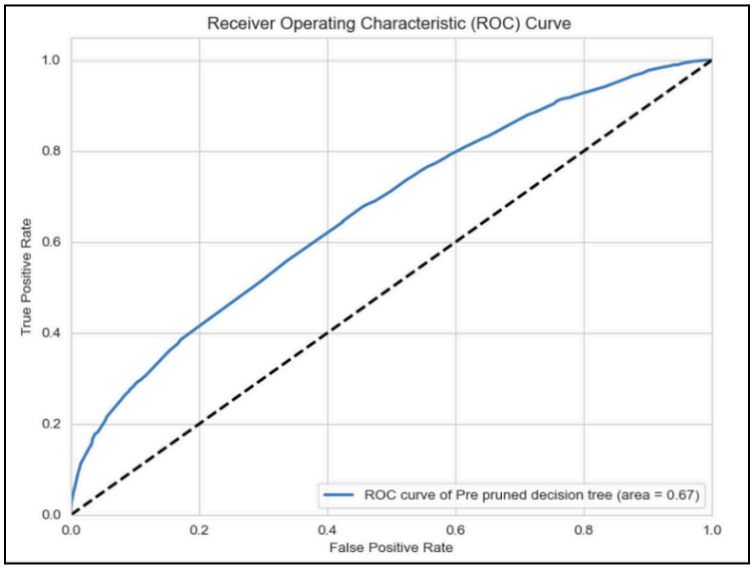


Figure 34

ROC curve of Pre-pruned decision tree



Post-pruned decision tree:

Post-pruning, also referred to as tree pruning or cost complexity pruning, is the process of removing or collapsing branches (subtrees) from a fully grown decision tree. Unlike pre-pruning, which applies constraints during the tree's construction, post-pruning is performed after the tree is completely built. The main goal of post-pruning is to reduce the tree's size, minimize overfitting, and enhance its ability to generalize to new data. The parameter ccp_alpha governs the tree's complexity, with higher values resulting in more aggressive pruning. The optimal alpha value determined is 0.00014.

Figure 35

Optimal alpha value of Post-pruned decision tree

```
Best alpha value of post pruned decision tree: 0.00014000098283027543
```

Figure 36

Accuracy vs. Alpha plot

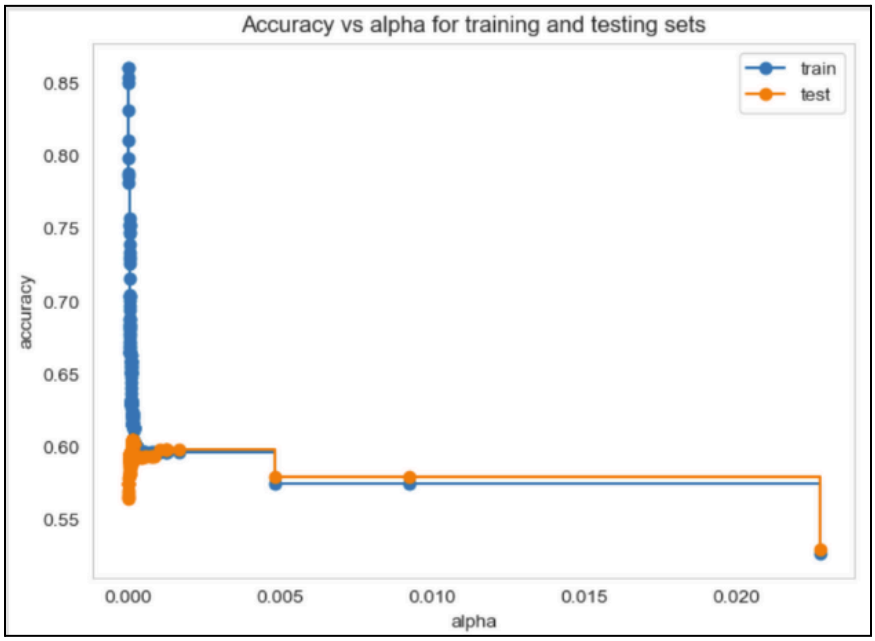


Figure 37

Various metrics of Post-pruned decision tree

Classifier	Accuracy	Confusion Matrix	Precision	Sensitivity or Recall	Specificity	F-score	AUC
Post pruned Decision Tree Classifier	0.57	[[2456 1334] [1751 1605]]	0.55	0.48	0.8	0.51	0.57

Figure 38

ROC curve of Post-pruned decision tree

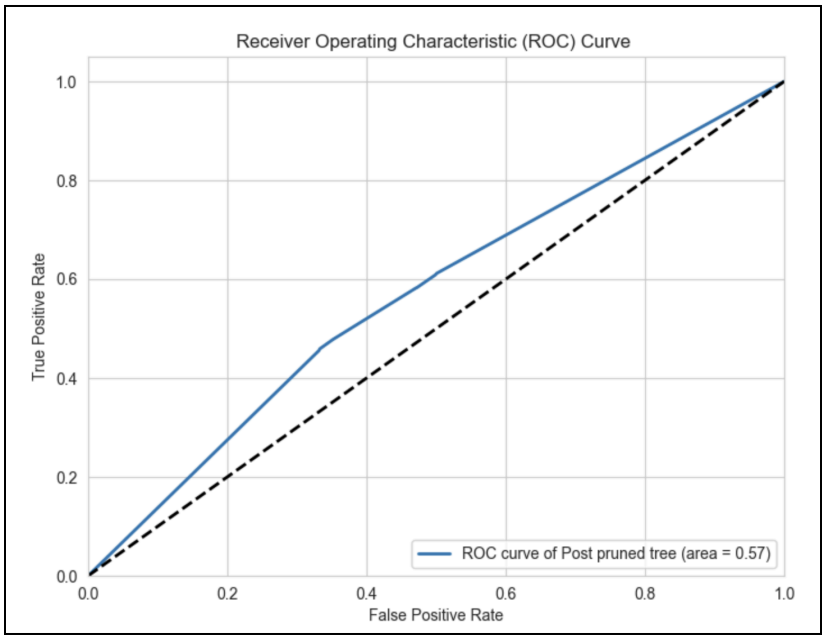
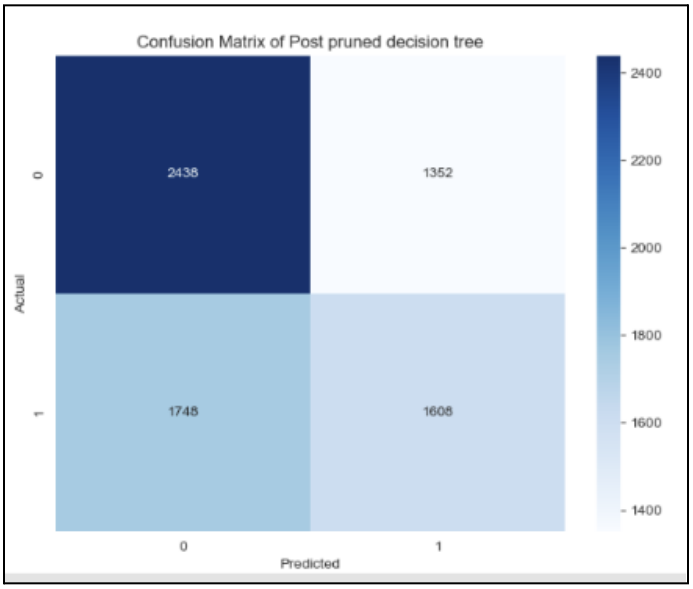


Figure 39

Confusion matrix of Post-pruned decision tree



Logistic Regression:

Logistic Regression is a statistical approach commonly applied to binary classification problems, where it predicts the probability of a categorical dependent variable, typically represented as 0 or 1. A grid search was conducted over the parameters 'C,' 'Solver,' and 'Penalty,' combined with stratified k-fold cross-validation, to identify the optimal hyperparameters.

Figure 40

Best hyperparameters of Logistic Regression

```
Best Hyperparameters: {'C': 0.1, 'penalty': 'l2', 'solver': 'saga'}
```

Figure 41

Various metrics of Logistic Regression

Classifier	Accuracy	Confusion Matrix	Precision	Sensitivity or Recall	Specificity	F-score	AUC
Logistic Regression	0.59	[[2872 918] [1947 1409]]	0.61	0.42	0.76	0.5	0.63

Figure 42

ROC curve of Logistic Regression

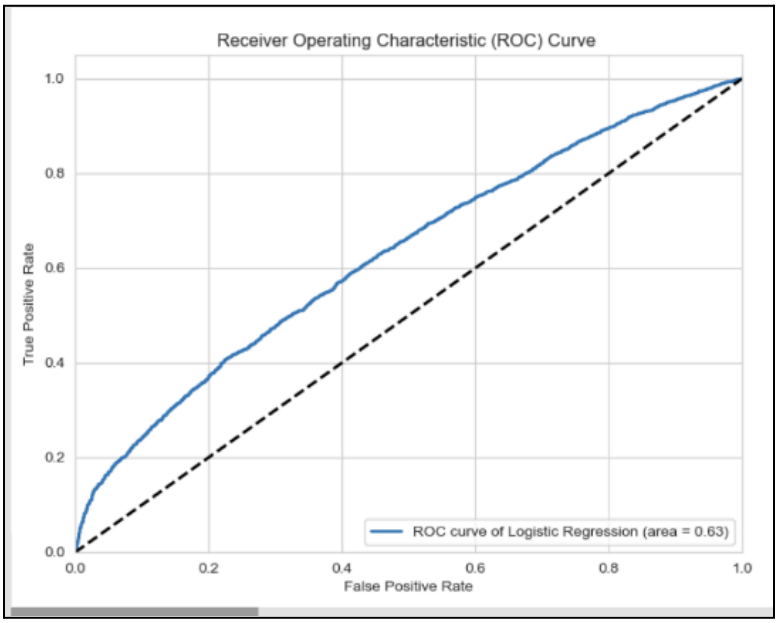
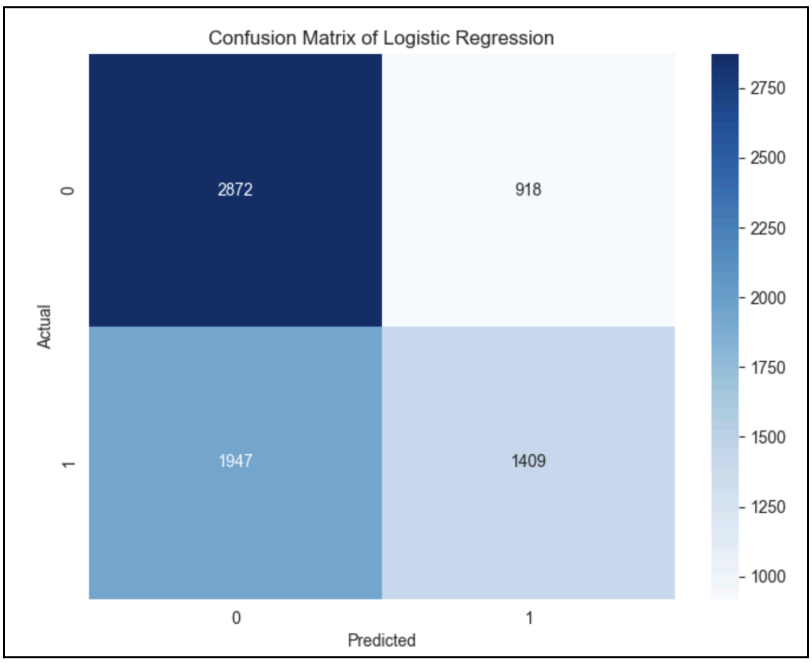


Figure 43

Confusion matrix of Logistic Regression



K Nearest Neighbors:

K-Nearest Neighbors (KNN) is a straightforward and efficient supervised learning algorithm applicable to both classification and regression tasks. The 'K' represents the number of nearest neighbors considered for making predictions. To classify or predict a new data point, the algorithm identifies its K closest neighbors using a specified distance metric. Grid search and stratified k-fold cross-validation were employed to determine the optimal hyperparameters. The best value of K, identified from the error rate vs. K graph, closely aligns with the grid search results.

Figure 44

Elbow method for finding optimal k

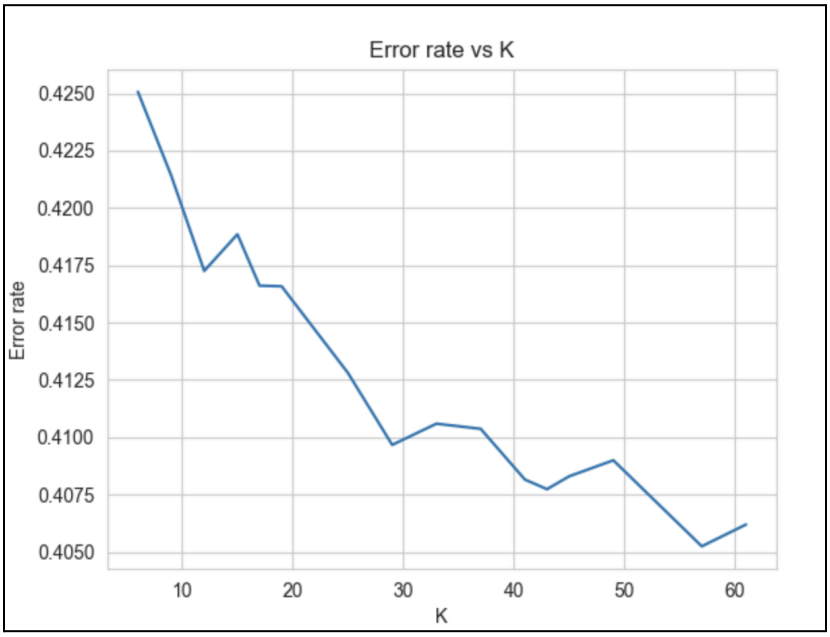


Figure 45

Best hyperparameters of K-Nearest Neighbors

```
Best Hyperparameters: {'n_neighbors': 57}
```

Figure 46

Various metrics of K-Nearest Neighbors

Classifier	Accuracy	Confusion Matrix	Precision	Sensitivity or Recall	Specificity	F-score	AUC
KNN	0.62	[[2714 1076] [1781 1575]]	0.59	0.47	0.72	0.52	0.63

Figure 47

ROC curve of K-Nearest Neighbors

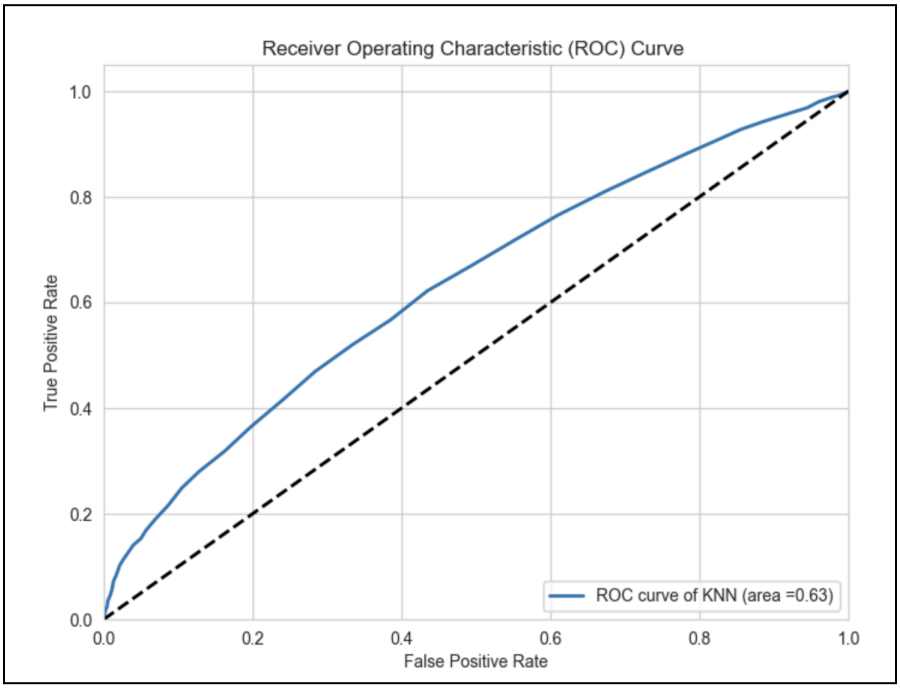
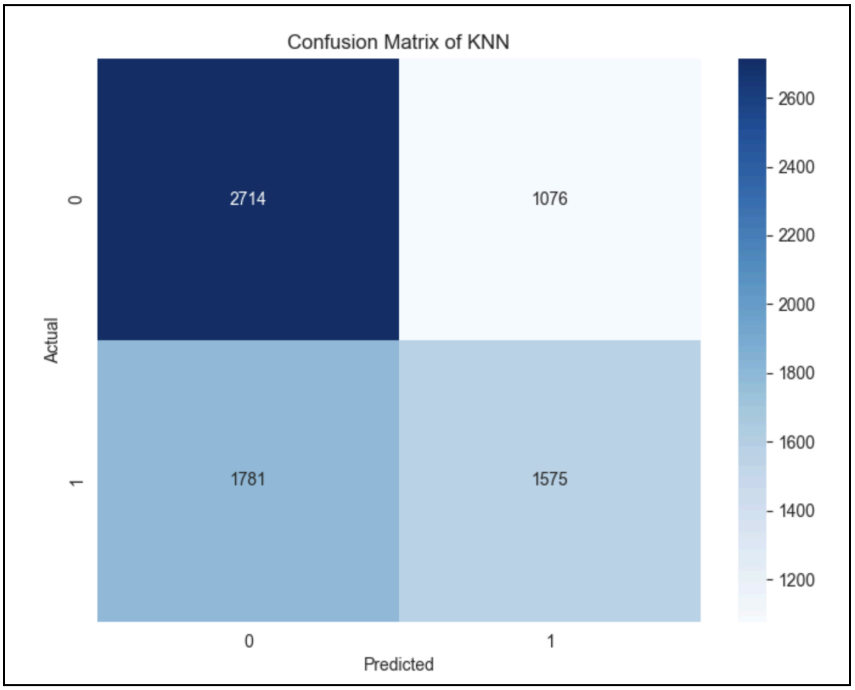


Figure 48

Confusion matrix of K-Nearest Neighbors



SVM:

Support Vector Machines (SVM) identify the hyperplane that maximizes the margin between the closest data points, known as support vectors, from different classes. A grid search was conducted with linear, polynomial, and RBF kernels, along with stratified k-fold cross-validation, to determine the optimal kernel. The radial basis function (RBF) kernel, commonly used for nonlinear problems, was found to be the best choice.

Figure 49

Best hyperparameters of SVM

```
Best Hyperparameters: {'kernel': 'rbf'}
```

Figure 50

Various metrics of SVM

Classifier	Accuracy	Confusion Matrix	Precision	Sensitivity or Recall	Specificity	F-score	AUC
SVM	0.6	[[3018 772] [2069 1287]]	0.63	0.38	0.8	0.48	0.63

Figure 51

Confusion matrix of SVM

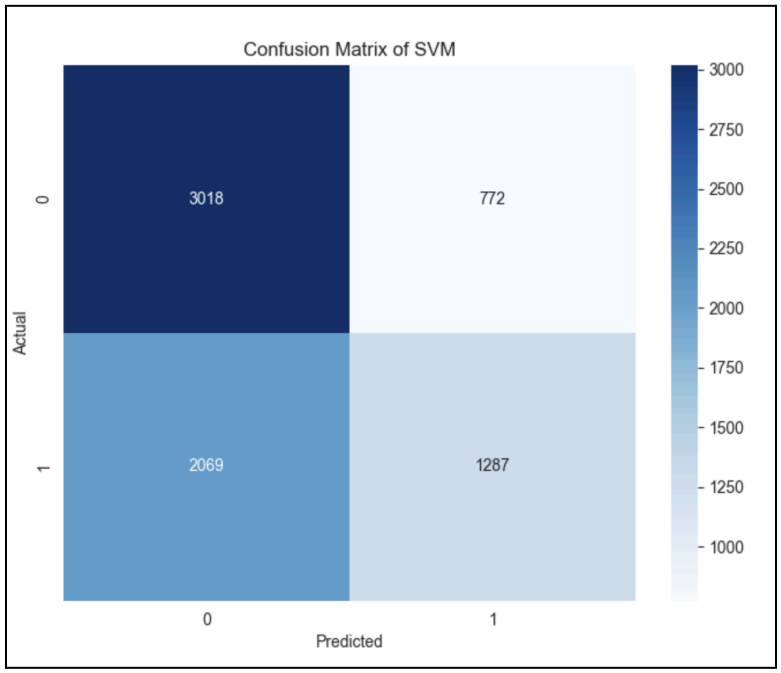
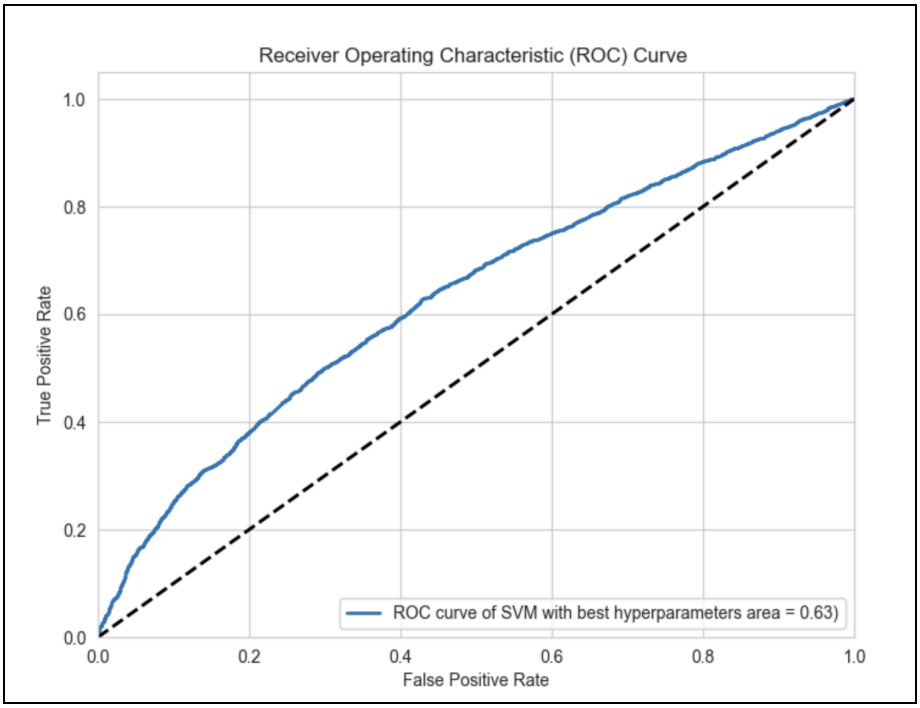


Figure 52

ROC curve of SVM



Naive Bayes:

Naive Bayes is a widely used and straightforward probabilistic algorithm primarily designed for classification tasks. It relies on Bayes' theorem and operates under the "naive" assumption that predictors (features) are independent of each other. Since the algorithm has no tunable parameters, grid search is not applicable. Instead, a stratified k-fold cross-validation approach was used to calculate the test accuracy.

Figure 53

Various metrics of Gaussian Naive Bayes

Classifier	Accuracy	Confusion Matrix	Precision	Sensitivity or Recall	Specificity	F-score	AUC
Gaussian Naive Bayes	0.58	[[3439 351] [2544 812]]	0.7	0.24	0.91	0.36	0.62

Figure 54

Confusion matrix of Gaussian Naive Bayes

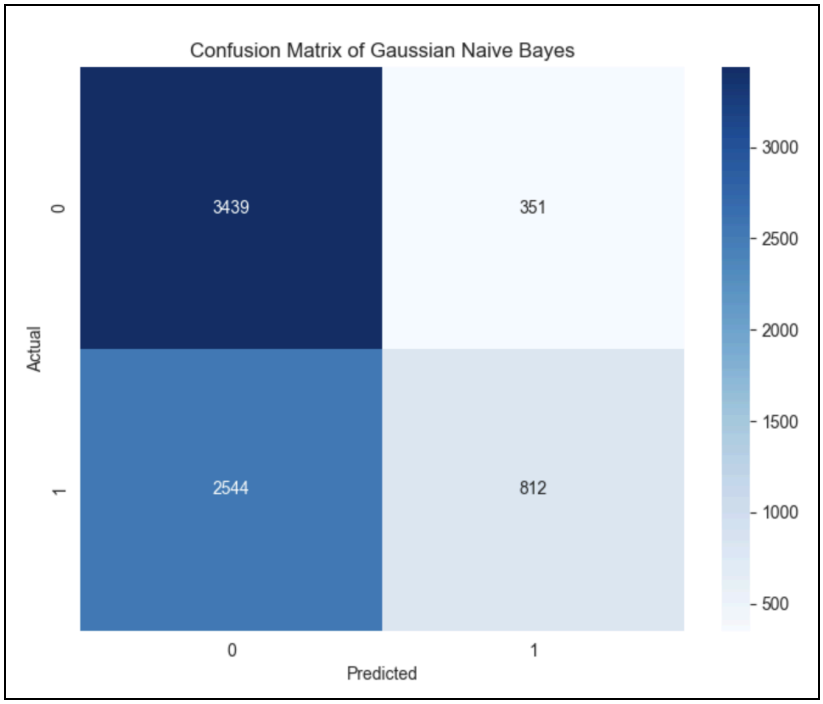
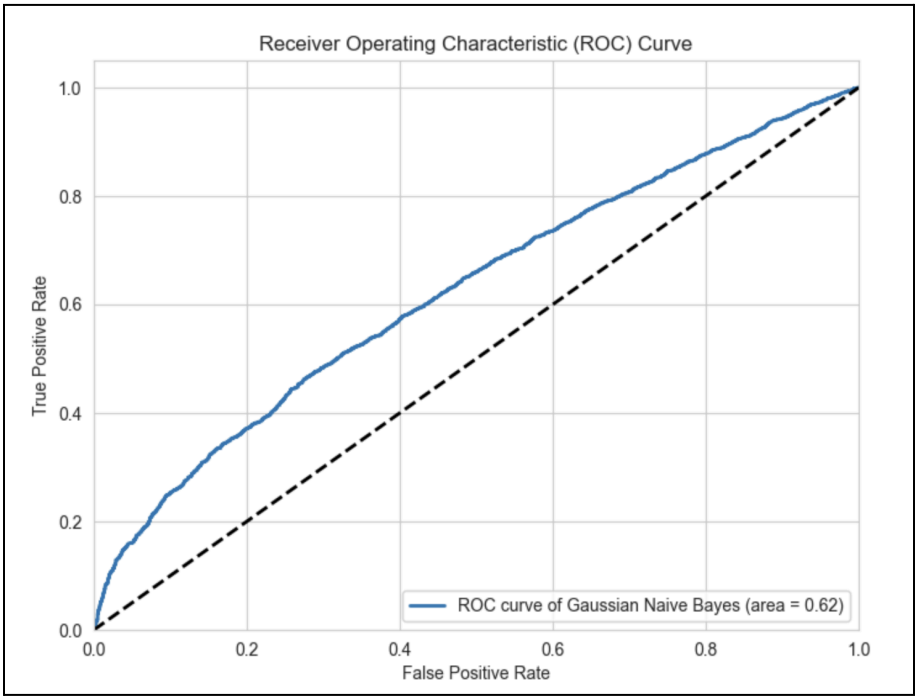


Figure 55

ROC curve of Gaussian Naive Bayes



Multi-layered Perceptron:

A Multi-Layer Perceptron (MLP) is a form of artificial neural network comprising multiple layers of neurons, capable of capturing complex relationships between input and output data. As a feedforward neural network, it maps input data to corresponding outputs. Grid search and stratified k-fold cross-validation were utilized to determine the optimal hyperparameters. The model demonstrates reasonable performance in terms of accuracy, precision, specificity, and AUC, but its recall and F-score are relatively low.

Figure 56

Best hyperparameters of the Multi-layered perceptron

```
Best Hyperparameters: {'hidden_layer_sizes': (15, 15)}
```

Figure 57

Various metrics of Multi-layered perceptron

Classifier	Accuracy	Confusion Matrix	Precision	Sensitivity or Recall	Specificity	F-score	AUC
Multilayer perceptron	0.61	[[2795 995] [1814 1542]]	0.61	0.46	0.74	0.52	0.64

Figure 58

Confusion matrix of Multi-layered perceptron

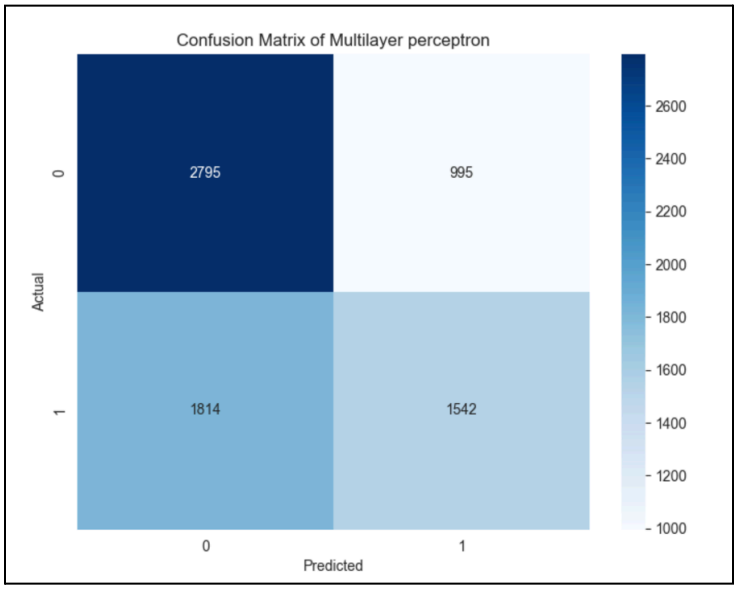
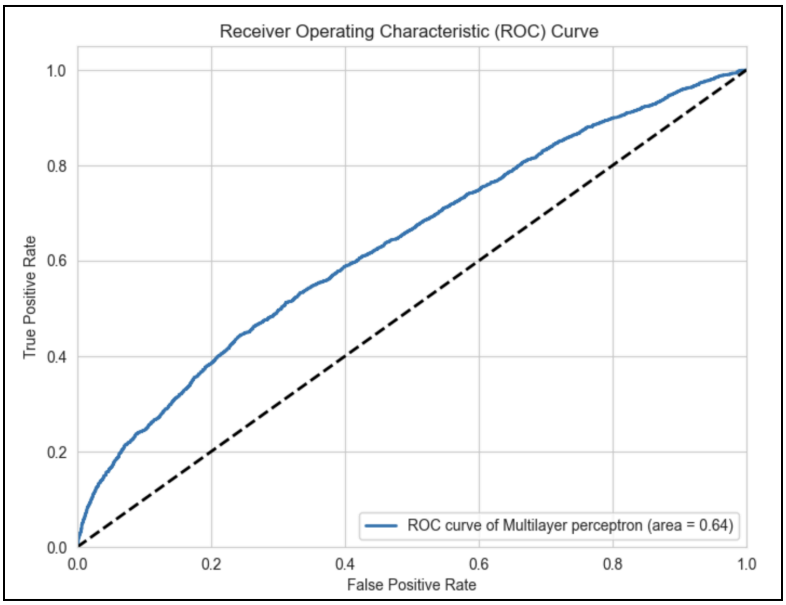


Figure 59

ROC curve of Multi-layered perceptron



Analysis of all classifiers of Phase 3:

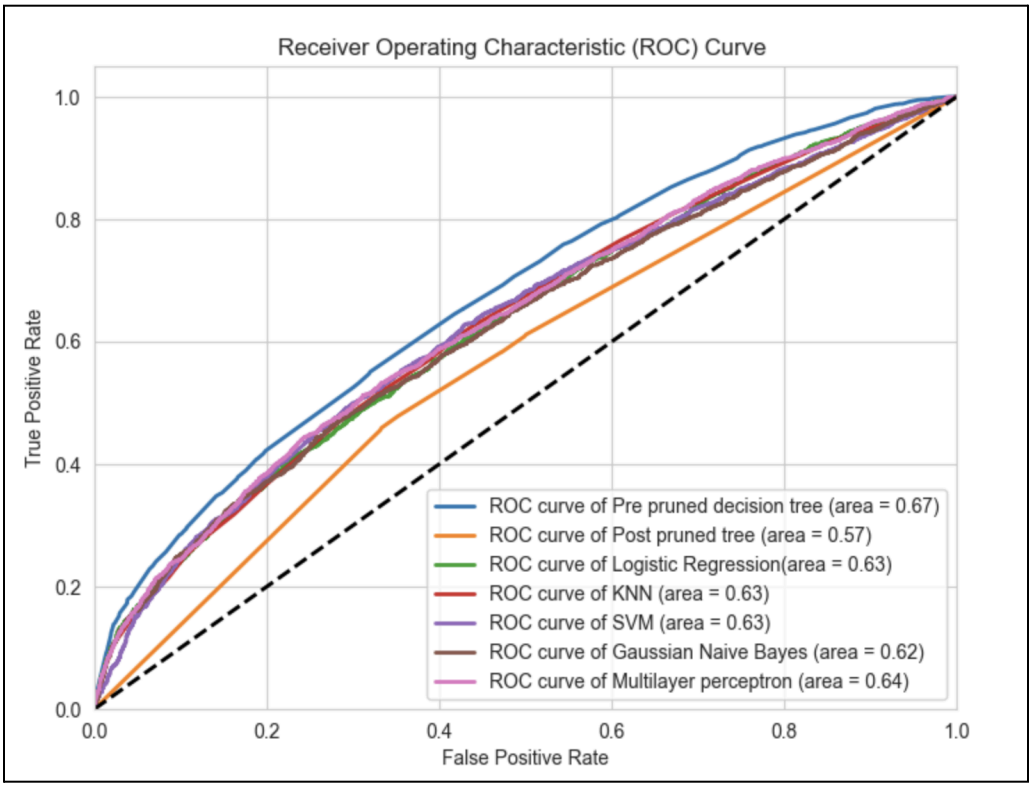
Figure 60

Comparison of all metrics of all classifiers

Classifier	Accuracy	Confusion Matrix	Precision	Sensitivity or Recall	Specificity	F-Score	AUC
Pre pruned Decision Tree Classifier	0.62	[[3047 743], [1956 1400]]	0.65	0.42	0.8	0.51	0.67
Post pruned Decision Tree Classifier	0.57	[[2456 1334], [1751 1605]]	0.55	0.48	0.8	0.51	0.57
Logistic Regression	0.6	[[2872 918], [1947 1409]]	0.61	0.42	0.76	0.5	0.63
KNN	0.6	[[2714 1076], [1781 1575]]	0.59	0.47	0.72	0.52	0.63
SVM	0.6	[[3018 772], [2069 1287]]	0.63	0.38	0.8	0.48	0.63
Gaussian Naive Bayes	0.58	[[3439 351], [2544 812]]	0.7	0.24	0.91	0.36	0.62
Multilayer Perceptron	0.61	[[2795 995], [1814 1542]]	0.61	0.46	0.74	0.52	0.64

Figure 61

ROC curve of all classifiers



Pre-pruned Decision Tree vs. Post-pruned Decision Tree: The pre-pruned decision tree has a slightly higher accuracy compared to the post-pruned one (0.62 vs. 0.57). However, the post-pruned tree shows better sensitivity/recall, indicating it's better at correctly identifying positive cases (delayed flights) after pruning. Post-pruning slightly improves the model's performance in correctly identifying delayed flights but results in a decrease in overall accuracy.

Logistic Regression, KNN, SVM, Gaussian Naive Bayes, and MLP: These models demonstrate similar accuracy scores around 0.58 to 0.61. KNN exhibits slightly better sensitivity (0.47) compared to others. Logistic Regression and SVM show decent precision (around 0.61 to 0.63) but moderate sensitivity (around 0.38 to 0.42).

Interpretation: Overall, ***The Pre-Pruned Decision Tree emerges as the best model*** due to its balanced performance across key metrics. It achieves the highest accuracy of 0.62, which is crucial for overall prediction reliability, and the highest AUC (0.67), reflecting its ability to distinguish between delayed and non-delayed flights effectively. Additionally, it exhibits strong precision (0.65), meaning it minimizes false positives while maintaining a reasonable sensitivity of 0.42. The model’s high specificity (0.80) indicates that it is highly effective in identifying non-delayed flights. Furthermore, its simplicity and interpretability make it an excellent choice for operational settings, such as in the airline industry, where decisions need to be both accurate and explainable.

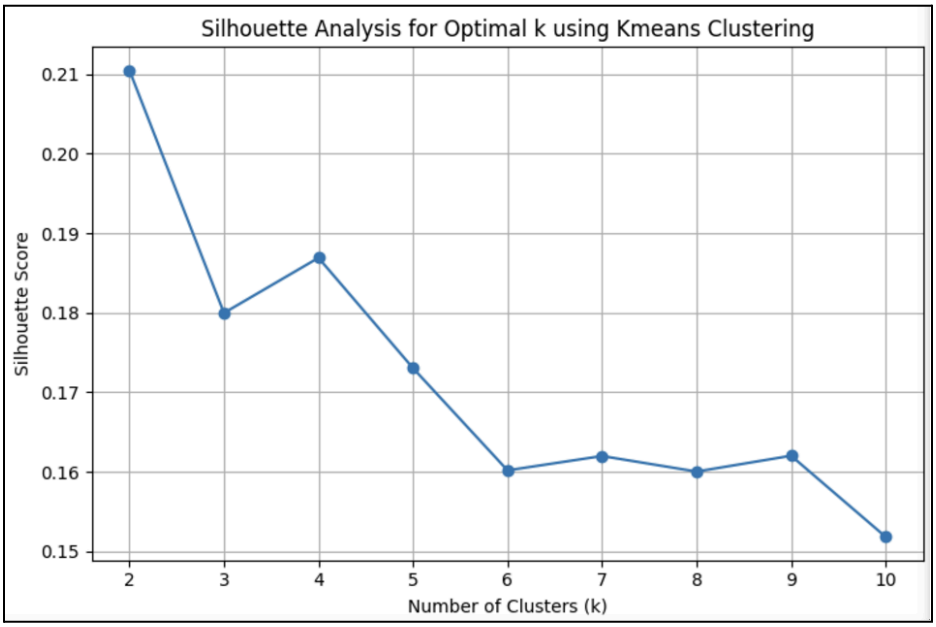
Phase IV: Clustering and Association

KMeans Clustering:

K-means clustering is a popular unsupervised machine learning algorithm used for clustering similar data points into groups or clusters. Silhouette Score measures how well-separated the clusters are. Silhouette analysis shows a higher score for k = 2, but this might happen because the algorithm is combining what should be multiple clusters into fewer larger clusters. A higher silhouette score does not always mean the best clustering if it oversimplifies the data. While the silhouette score for k = 2 is higher, the WCSS elbow at k = 4 suggests that the dataset likely has four natural groupings. The silhouette score at k = 4 is still acceptable, and the drop in silhouette score after k = 4 further validates this choice. Therefore, **optimal value k is 4**.

Figure 62

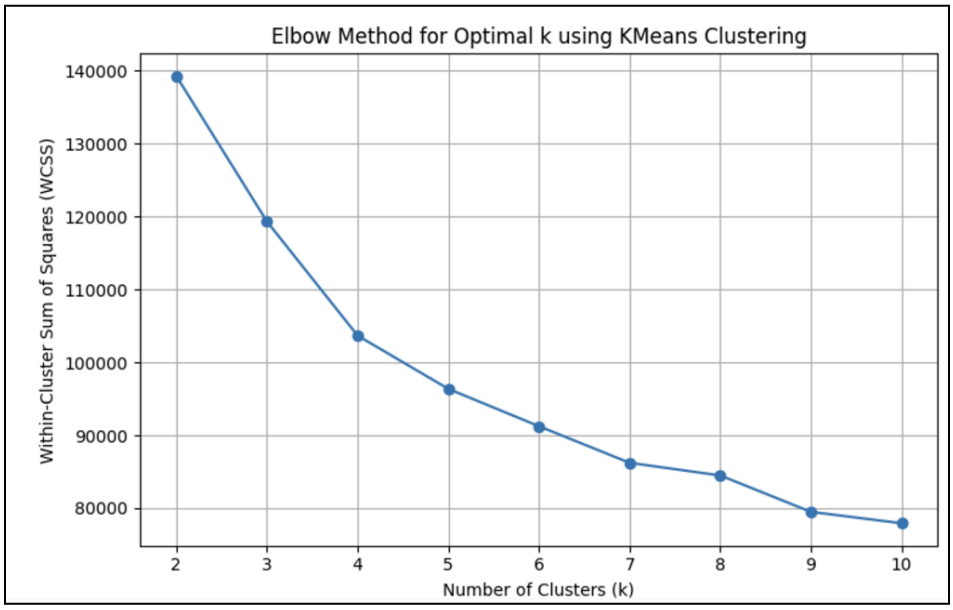
Silhouette analysis for Optimal k using KMeans clustering



The silhouette score reaches its highest value at k = 2, suggesting the data can be grouped into two broad clusters. However, another significant peak is observed at k = 4, which also represents a well-separated clustering structure.After k = 4, the silhouette score consistently decreases, indicating that further increasing the number of clusters does not meaningfully improve the clustering quality. This choice of K =4 is also supported by the elbow plot that is plotted and discussed below.

Figure 63

WCSS plot analysis for optimal k using KMeans clustering



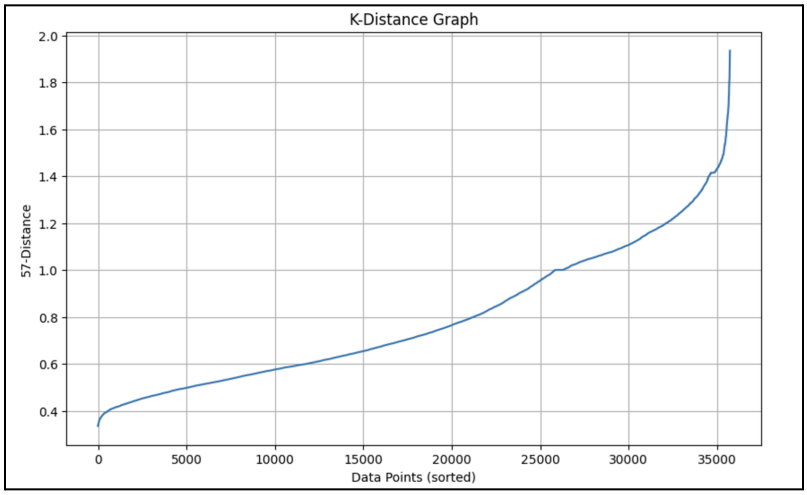
Within-Cluster Sum of Squares (WCSS): Measures the compactness of clusters. Lower WCSS implies more compact clusters. The elbow point is where the rate of decrease in within-cluster variation slows down significantly, forming an “elbow” shape. In this graph, the elbow appears to occur around k = 4, as the reduction in within-cluster variation beyond this point becomes more gradual. Thus, k = 4 is likely the optimal number of clusters based on this plot.

DBSCAN algorithm:

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm that identifies clusters of high-density points while labeling sparser points as noise. It does not require specifying the number of clusters in advance, making it ideal for datasets with arbitrarily shaped clusters. DBSCAN relies on two key parameters: eps (radius of neighborhood) and min_samples (minimum points to form a dense cluster). These are determined using the K-Distance graph, where the "elbow point" indicates a suitable eps.

Figure 64

K-distance graph for optimal eps for dbscan algorithm




```
Number of clusters: 5
Number of noise points: 17
Silhouette Score: 0.05707403431565488
```

In this analysis, the K-Distance graph suggested an eps of approximately 0.5 with min_samples = 5. DBSCAN identified 5 clusters and 17 noise points, achieving a Silhouette Score of 0.057, which indicates weak separation between clusters. This low score might result from overlapping clusters or varying densities within the data. DBSCAN effectively identifies noise and handles non-spherical clusters but can be sensitive to parameter selection and struggles with datasets having highly variable cluster densities.

Apriori algorithm:

The Apriori algorithm is a classic algorithm used in association rule mining, particularly for discovering frequent item sets in transactional databases or datasets. The support for flights labeled as "Large" is 18.97%, while for "Short" lengths, it's 41.42%. There's an interesting relationship here between flights labeled as "Small" and specific conditions related to both "Medium" length and the "Start of the Week" for the day of the week

Figure 65

Rules generated from Apriori

Processing 4 combinations | Sampling itemset size 4

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
36	(Large Aircraft)	(Short Length)	0.18	0.45	0.12	0.65	1.45	0.04	1.59	0.38
64	(Average Length, StartOfWeek)	(Small Aircraft)	0.18	0.51	0.10	0.56	1.10	0.01	1.12	0.12
9	(Average Length)	(Small Aircraft)	0.43	0.51	0.24	0.56	1.10	0.02	1.12	0.17
3	(Average Length)	(Early Morning)	0.43	0.52	0.23	0.55	1.05	0.01	1.06	0.09
58	(Early Morning, Average Length)	(Small Aircraft)	0.23	0.51	0.13	0.54	1.08	0.01	1.08	0.09

The presence of "Large Aircraft" (18% of occurrences) is associated with "Short Length" (45% of occurrences) in 12% of cases. The confidence is moderate at 65%, indicating that when there's a large aircraft, there's a 65% chance that the flight will be of short length.

The lift value of 1.45 implies that the occurrence of a large aircraft increases the chance of a short-length flight by 45% compared to what would be expected if the two were independent.

When flights start at the beginning of the week and have an average length, there's a 56% chance they'll involve small aircraft. The lift value of 1.10 suggests a marginal increase in the likelihood of small aircraft in these situations. Flights with an average length and scheduled in the early morning have a 55% chance of occurrence. The lift value is close to 1, indicating no significant association beyond what would be expected by chance. When flights are early in the morning, of average length, there's a 54% chance they'll involve small aircraft. Both the lift and confidence are close to 1, suggesting a mild association without a significant deviation from chance.

Recommendations

This project involved the practical application of diverse machine learning techniques on an airline dataset to predict flight delays. Throughout the process, I gained valuable experience in implementing various models, fine-tuning hyperparameters, and identifying optimal classifiers. This hands-on exploration not only strengthened my ability to work with real-world datasets but also highlighted the importance of efficient training and hyperparameter optimization in achieving robust model performance.

Looking ahead, I plan to enhance this project by integrating the dataset with additional sources, such as flight schedules, to provide more contextual information. I also aim to experiment with advanced models like XGBoost to further improve prediction accuracy and performance.

Among the evaluated models, ***The Pre-Pruned Decision Tree emerges as the best model*** due to its balanced performance across key metrics. It achieves the highest accuracy of 0.62, which is crucial for overall prediction reliability, and the highest AUC (0.67). In terms of feature importance, attributes such as Time, Length, Flight, and Airline_WN showed a strong relationship with the target variable, Delay. The prominence of Airline_WN, which represents the busiest airline in the dataset, likely contributes to higher delays. Additionally, the type of aircraft used also appeared to influence flight delays significantly.

Finally, the KMeans clustering analysis identified four clusters as optimal, While k = 2 achieves the highest silhouette score, k = 4 aligns better with the Elbow Method and provides a more detailed segmentation of the data. This suggests that the dataset contains moderate differentiation into four meaningful subgroups, despite some overlapping characteristics.

Appendix

Importing necessary libraries:

```
#%%----------Phase I: Feature Engineering & EDA--------
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split,StratifiedKFold, GridSearchCV
from sklearn.ensemble import RandomForestClassifier,StackingClassifier
from sklearn.decomposition import PCA,TruncatedSVD
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.feature_selection import f_classif
from prettytable import PrettyTable
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error, accuracy_score, confusion_matrix,roc_auc_score, roc_curve
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
```

Data Cleaning/ Check for data duplications and removal:

```
#Data cleaning
df = pd.read_csv(r'/Users/jyothi/Documents/SEM2/Machine_Learning/Project/Airlines.csv')
print(df.columns)
print(df.head().to_string())
print("\nMissing values in the dataset")
print(df.isna().sum().sum() + df.isnull().sum().sum())

#Check for data duplications and removal
print("\nDuplicated values in the dataset")
print(df.duplicated().sum())
print("\nNumber of unique values in each feature of whole dataset")
print(df.nunique())
```

Aggregation and Downsampling:

```
#Aggregation
df_dayofweek = df[['DayOfWeek', 'Delay']].groupby('DayOfWeek')['Delay'].sum().reset_index().sort_values(by='Delay', ascending=False)
print(df_dayofweek.to_string())
df_airline = df[['Airline', 'Delay']].groupby('Airline')['Delay'].sum().reset_index().sort_values(by='Delay', ascending=False)
print(df_airline.to_string())

#Down sampling
df = df.drop( labels: ['id', 'AirportTo'], axis=1)
df_airline = df['Airline'].value_counts().sort_values(ascending=False)
df = df[df['Airline'].isin(['WN', 'DL', 'OO', 'AA', 'MQ'])]
df_airport_to = df['AirportFrom'].value_counts().sort_values(ascending=False)
df = df[df['AirportFrom'].isin(['ATL', 'ORD', 'DFW', 'DEN', 'LAX'])]
df = df[::2]
print("\nThe shape of the dataset after downsampling:")
print(df.shape)
```

Dimensionality reduction/feature selection:

Using Random Forest Analysis:

```
# Dimensionality reduction
# Feature importance using Random forest analysis
model_rf = RandomForestClassifier(random_state=5805)
model_rf.fit(X, y)
importances = model_rf.feature_importances_
print("Feature importances from Random forest method")
indices = np.argsort(importances)
sortedImportance = importances[indices] * 100
print(sortedImportance[::-1].round(2))
sorted_features = X.columns[indices]
plt.figure(figsize=(15, 10))
plt.barh(range(len(sortedImportance)), sortedImportance, color='darkblue')  # Light blue hex code
plt.yticks(range(len(sorted_features)), sorted_features)
plt.title('Feature importances by RandomForestClassifier')
plt.xlabel('Feature Importance')
plt.ylabel('Features')
plt.show()
```

Using Principal Component Analysis:

```
# Dimensionality reduction/Feature selection
# Using PCA
pca = PCA()
pca.fit(X)
cum_var = np.round(np.cumsum(sorted(pca.explained_variance_ratio_, reverse=True)) * 100, decimals=2)
print("Cumulative explained variance of each component from PCA")
print(cum_var)
labels = [i for i in range(1, len(cum_var) + 1)]
plt.plot( *args: range(len(cum_var)), cum_var, alpha=0.9, color='darkblue')  # Dark blue line
x_point = 11
y_point = 95
plt.axvline(x=x_point, color='darkblue', linestyle='--')
plt.axhline(y=y_point, color='darkblue', linestyle='--')
plt.annotate( text: f'({x_point},{y_point})', xy: (x_point, y_point), color='black', textcoords="offset points", xytext=(-15, 10), ha='center')
plt.xlabel( xlabel: "Number of components", color='black')
plt.ylabel( ylabel: "Percentage contribution to variance", color='black')
plt.title( label: "Cumulative explained variance of each component from PCA", color='black')
plt.tick_params(axis='x', colors='black')
plt.tick_params(axis='y', colors='black')
plt.show()
```

Using Singular Value Decomposition Analysis:

```
# Dimensionality reduction/Feature selection
# Using Singluar Value Decomposition
tsvd = TruncatedSVD(n_components=17)
tsvd_result = tsvd.fit(X)
print("Explained variance ratio from SVD ", tsvd.explained_variance_ratio_.round(2))
print(f"Singular values of features {tsvd.singular_values_.round(2)}")
cum_evr = np.cumsum(100 * tsvd.explained_variance_ratio_)
labels = [i for i in range(1, len(cum_evr) + 1)]
plt.bar(x=labels, height=cum_evr, alpha=0.8, color='darkblue')
x_point = 12
y_point = 95
plt.axvline(x=x_point, linestyle='--', color='red')
plt.axhline(y=y_point, linestyle='--', color='red')
plt.annotate( text: f'({x_point},{y_point})', xy: (x_point, y_point), textcoords="offset points", xytext=(-15,10), color='black')
plt.xlabel( xlabel: "Number of components", color='black')
plt.ylabel( ylabel: "Percentage contribution to variance", color='black')
plt.title( label: "Cumulative variance ratio of each component from SVD", color='black')
plt.xticks(labels)
plt.show()
```

Using VIF and condition number:

```
# Dimensionality reduction/Feature selection
# Using Variance Inflation Factor
print("VIF analysis")
vif_fea = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
for i,j in zip(df.columns, vif_fea):
    print(i,round(j,2))
_, S, _ = np.linalg.svd(X, full_matrices=False)
print("Singular values of the dataset")
print(S.round(2))
condition_number = np.linalg.cond(X_train)
print("Condition number : ", round(condition_number,2))

# Data imbalance analysis
sns.set_style('whitegrid')
sns.countplot(data=df,x='Delay')
plt.title('Countplot of # of obs. for each class of Delay')
plt.xlabel("Whether flight delayed")
plt.ylabel("# of samples")
plt.show()
```

Discretization & Binarization: Label Encoding/one hot encoding :

```
#Discretization & Binarization: Label Encoding/one hot encoding
one_hot_enc_col = ['AirportFrom', 'Airline', 'DayOfWeek']
df = pd.get_dummies(df, columns=one_hot_enc_col, drop_first=True)
for i in df.columns:
    if i not in ['Length', 'Time', 'Delay', 'Flight']:
        df[i] = df[i].map({False:0, True:1})
print("\nOne hot encoded dataset")
print(df.head().to_string())
df_orig= df[::]
X_orig = df.drop( labels: 'Delay', axis=1)
X_reg = df[::]
```

Variable Transformation: Standardization:

```
#Variable Transformation: Standardization
sc = StandardScaler()
for i in ['Flight', 'Length', 'Time']:
    df[i] = sc.fit_transform(df[[i]])
print("\nStandardized dataset")
print(df.head().to_string())
X = df.drop( labels: 'Delay', axis=1)
y = df['Delay']
print(X.shape)
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.3,random_state=5805, shuffle=True,stratify=y)
```

Outlier Analysis and Removal:

```
#Anomaly detection/Outlier Analysis and removal using IQR method
q1 = df['Length'].quantile(0.25)
q3 = df['Length'].quantile(0.75)
IQR = q3 - q1
lower_bound = q1 - 1.5 * IQR
upper_bound = q3 + 1.5 * IQR
df = df[(df['Length'] < upper_bound) & (df['Length'] > lower_bound)]
df_new = df[::]
sns.boxplot(data=df, x='Length')
plt.title("Box plot of Length feature after removal of outliers using IQR method")
plt.show()
```

Calculating and plotting of correlation matrix:

```
# Correlation matrix using heatmap
correlation = df.corr().round(2)
plt.figure(figsize = (18,11))
sns.heatmap(correlation, annot = True, cbar=True)
plt.title('Heatmap of correlation between all the features')
plt.show()
```

Calculating and plotting of covariance matrix:

```
# Covariance matrix using heatmap
correlation = df.cov().round(2)
plt.figure(figsize = (18,11))
sns.heatmap(correlation, annot = True, cbar=True)
plt.title('Heatmap of covariance between all the features')
plt.show()
```

T-test analysis:

```
#T-test analysis
pt_ttest = PrettyTable()
pt_ttest.field_names = ['Feature', 't-statistic', 'p-value']
print("t-test analysis")
for i in range(len(X_reg_train.columns)):
    pt_ttest.add_row([X_reg_train.columns[i], round(model_ols.tvalues[i],2),round(model_ols.pvalues[i],2)])
print(pt_ttest)
```

F-test analysis:

```
#F-test analysis
f_test, p_values = f_classif(X_reg, y_reg)
pt = PrettyTable()
pt.field_names = ['Feature', 'F-test score', 'p-value']
for i in range(len(f_test)):
    pt.add_row([X.columns[i], round(f_test[i],2), round(p_values[i],2)])
print(pt)
```

Linear regression:

```
# Prediction with Linear Regression
model_linear = LinearRegression()
X_reg_train = sm.add_constant(X_reg_train)
X_reg_test = sm.add_constant(X_reg_test)
model_linear.fit(X_reg_train, y_reg_train)
y_pred_train = model_linear.predict(X_reg_train)
y_pred_train_orig = y_pred_train*std_length+mean_length
y_pred_test = model_linear.predict(X_reg_test)
y_pred_test_orig = y_pred_test*std_length+mean_length
print("The coefficients of the Linear Regression model")
print("Feature", "Coefficient")
for i in range(len(X_reg_train.columns)):
    print(X_reg_train.columns[i], round(model_linear.coef_[0][i],2))
mse = mean_squared_error(y_reg_train*std_length+mean_length, y_pred_train_orig)
print(f"Mean Squared Error for training set (MSE): {mse:.3f}")
pt=PrettyTable()
pt.field_names=['Model', 'R-squared', 'Adj. R-squared', 'AIC', 'BIC', 'Mean Squared Error']
r_squared = model_linear.score(X_reg_train, y_reg_train)
n = len(y_reg_train)
p = X_reg_train.shape[1] - 1
adjusted_r_squared = 1 - (1 - r_squared) * (n - 1) / (n - p - 1)
aic = n*np.log(mse) + 2*p
bic = n*np.log(mse) + p*np.log(n)
pt.add_row(['Linear Regression', round(r_squared,2), round(adjusted_r_squared,2),
round(aic,2), round(bic,2), round(mse,2)])
print(pt)
index=np.arange(len(y_reg_test))
plt.title("Prediction with Linear Regression")
plt.plot( *args: index, y_reg_test * std_length + mean_length, color='blue', label='Actual values of test set')  # Blue for actual values
plt.plot( *args: index, y_pred_test_orig, color='lightyellow', label='Predicted values with test set')  # Yellow for predicted values
plt.xlabel("# of samples")
plt.ylabel("Length")
plt.legend()
plt.show()
```

Confidence Interval analysis:

```
#Confidence interval analysis
index=np.arange(len(y_reg_test))
y_prediction = model_ols.get_prediction(X_reg_test)
conf_intr = model_ols.conf_int()
ci_pred_frame=y_prediction.summary_frame(alpha=0.05)
ci_upper = ci_pred_frame.obs_ci_upper
ci_lower = ci_pred_frame.obs_ci_lower
plt.plot( *args: index,y_pred_test_orig,label="Predicted",color='blue')
plt.fill_between(index, ci_upper*std_length+mean_length,
ci_lower*std_length+mean_length, color='blue', alpha=0.3, label="Confidence Interval")
plt.xlabel("Number of observations")
plt.ylabel("Sales")
plt.title("Confidence interval of Predicted values summary_frame")
plt.legend()
plt.show()
model_ols_sr = sm.OLS(y_reg_train, X_reg_train).fit()
print(model_ols_sr.summary())
```

Backward stepwise regression:

```
## Backward stepwise regression
print("Eliminating the feature with highest p-value which is DayOfWeek_3")
X_reg_train = X_reg_train.drop('DayOfWeek_3', axis=1)
X_reg_test = X_reg_test.drop('DayOfWeek_3', axis=1)
model_ols = sm.OLS(y_reg_train, X_reg_train).fit()
y_pred_train = model_ols.predict(X_reg_train)
y_pred_train_orig = y_pred_train*std_length+mean_length
y_pred_test = model_ols.predict(X_reg_test)
y_pred_test_orig = y_pred_test*std_length+mean_length
print(model_ols.summary())

print("Eliminating DayOfWeek_4")
X_reg_train = X_reg_train.drop('DayOfWeek_4', axis=1)
X_reg_test = X_reg_test.drop('DayOfWeek_4', axis=1)
model_ols = sm.OLS(y_reg_train, X_reg_train).fit()
y_pred_train = model_ols.predict(X_reg_train)
y_pred_train_orig = y_pred_train*std_length+mean_length
y_pred_test = model_ols.predict(X_reg_test)
y_pred_test_orig = y_pred_test*std_length+mean_length
print(model_ols.summary())
```

Pre-pruned decision tree:

```
## Pre-prune decision tree
X = df.drop( labels: 'Delay', axis=1)
print(X.columns)
y = df['Delay']
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2,random_state=5805, shuffle=True)
pt = PrettyTable()
pt.field_names = ['Classifier', 'Accuracy', 'Confusion Matrix', 'Precision','Sensitivity or Recall', 'Specificity', 'F-score', 'AUC']
clf=DecisionTreeClassifier(random_state=5805)
tuned_parameters = [{'max_depth':[5,7,9,11],
'min_samples_split': [2,5,10,15],
'min_samples_leaf':[1,2,4,8],
'max_features':[2,4,7],
'splitter':['best','random'],
'criterion':['gini','entropy','log_loss']}]
stratified_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=5805)
grid_search = GridSearchCV(clf, tuned_parameters, cv=stratified_kfold,scoring='accuracy')
grid_search.fit(X, y)
print("Best Hyperparameters:", grid_search.best_params_)
best_classifier = grid_search.best_estimator_
y_train_pred = best_classifier.predict(X_train)
y_test_pred = best_classifier.predict(X_test)
pre_acc = accuracy_score(y_test, y_test_pred)
print(f'Train accuracy of pre pruned decision tree {round(accuracy_score(y_test,y_test_pred),2)}')
print(f'Test accuracy of pre pruned decision tree {round(pre_acc, 2)}')
conf_matrix_pre = confusion_matrix(y_test, y_test_pred)
tn, fp, fn, tp = conf_matrix_pre.ravel()
specificity_pre = tn/(tn+fp)
y_prob_pre = best_classifier.predict_proba(X_test)[:, 1] # Probability of the positive class
roc_auc_pre = roc_auc_score(y_test, y_prob_pre)
recall_pre = recall_score(y_test, y_test_pred)
precision_pre = precision_score(y_test, y_test_pred)
f1_pre = f1_score(y_test, y_test_pred)
# heatmap of confusion matrix
plt.figure(figsize=(8, 6))
```

Post-pruned decision tree:

```
## Post pruned decision tree
fold_accuracies = []
fold_train_accuracies = []
path = best_classifier.cost_complexity_pruning_path(X_train,y_train)
alphas = path['ccp_alphas']

# Iterate over different values of ccp_alpha
for alpha in alphas:
    clf = DecisionTreeClassifier(random_state=5805, ccp_alpha=alpha)
    clf.fit(X_train, y_train)
    accuracy_test = clf.score(X_test, y_test)
    fold_accuracies.append(accuracy_test)
    accuracy_train = clf.score(X_train, y_train)
    fold_train_accuracies.append(accuracy_train)

best_alpha_index = np.argmax(fold_accuracies)
best_alpha = alphas[best_alpha_index]
print(f"Best alpha value of post pruned decision tree: {best_alpha}")

# Plot accuracy vs. alpha for train and test sets
fig, ax = plt.subplots()
ax.set_xlabel('alpha')
ax.set_ylabel('accuracy')
ax.set_title("Accuracy vs alpha for training and testing sets")
ax.plot( *args: alphas, fold_train_accuracies, marker="o", label="Train Accuracy", drawstyle="steps-post")
ax.plot( *args: alphas, fold_accuracies, marker="o", label="Test Accuracy", drawstyle="steps-post")
ax.legend()
plt.grid()
plt.tight_layout()
plt.show()
best_alpha = alphas[np.argmax(accuracy_test)]
print("Best alpha value of post pruned decision tree", best_alpha)

clf = DecisionTreeClassifier(random_state=5805, ccp_alpha=best_alpha)
clf.fit(X_train, y_train)
y_train_pred = clf.predict(X_train)
```

Logistic regression:

```
## Logistic Regression
param_grid = {
'C': [0.1, 1.0, 10.0],
'penalty': ['l1', 'l2'],
'solver': ['liblinear', 'saga']
}
model_log = LogisticRegression(random_state=5805, max_iter=400)
grid_search = GridSearchCV(model_log, param_grid, cv=stratified_kfold,
scoring='accuracy')
grid_search.fit(X, y)
print("Best Hyperparameters:", grid_search.best_params_)
model_log = grid_search.best_estimator_
model_log.fit(X_train, y_train)
y_train_pred = model_log.predict(X_train)
y_test_pred = model_log.predict(X_test)
log_acc_train = accuracy_score(y_train, y_train_pred)
log_acc_test = accuracy_score(y_test, y_test_pred)
print(f'Train accuracy of Logistic Regression {round(accuracy_score(y_train,y_train_pred),2)}')
print(f'Test accuracy of Logistic Regression {round(log_acc_test,2)}')
conf_matrix_log = confusion_matrix(y_test, y_test_pred)
tn, fp, fn, tp = conf_matrix_log.ravel()
specificity_log = tn/(tn+fp)
y_prob_log = model_log.predict_proba(X_test)[:, 1]
roc_auc_log = roc_auc_score(y_test, y_prob_log)
recall_log = recall_score(y_test, y_test_pred)
precision_log = precision_score(y_test, y_test_pred)
f1_log = f1_score(y_test, y_test_pred)
pt2 = PrettyTable()
pt2.field_names = ['Classifier', 'Accuracy', 'Confusion Matrix', 'Precision',
'Sensitivity or Recall', 'Specificity', 'F-score', 'AUC']
pt2.add_row(['Logistic Regression', round(np.mean(log_acc_train),2),
```

KNN classifier:

```
## KNN Classifier
param_grid = {'n_neighbors': [6, 9, 12, 15, 17, 19, 25, 29, 33, 37, 41, 43, 45, 49, 53, 57,61],
}
model_knn = KNeighborsClassifier()
grid_search = GridSearchCV(model_knn, param_grid, cv=stratified_kfold,scoring='accuracy')
grid_search.fit(X, y)
results = grid_search.cv_results_
k_values = results['param_n_neighbors'].data
mean_test_scores = results['mean_test_score']
error_rate = 1-mean_test_scores
plt.plot( *args: k_values, error_rate)
plt.xlabel('K')
plt.ylabel('Error rate')
plt.title('Error rate vs K')
plt.show()
print("Best Hyperparameters:", grid_search.best_params_)
```

SVM classifier:

```
## SVM Classifier
parameters = {
'kernel': ['linear', 'rbf', 'poly'], # Kernel types to try
}
# Perform grid search with cross-validation
svm_classifier = SVC(probability=True)
grid_search = GridSearchCV(svm_classifier, parameters, cv=stratified_kfold,n_jobs=-1)
grid_search.fit(X, y)
print("Best Hyperparameters:", grid_search.best_params_)

svm_lr = grid_search.best_estimator_
svm_lr.fit(X_train, y_train)
y_train_pred = svm_lr.predict(X_train)
y_test_pred = svm_lr.predict(X_test)
svm_lr_acc = accuracy_score(y_test, y_test_pred)
print(f'Train accuracy of SVM with best hyperparameters {round(np.mean(svm_lr_acc),2)}')
print(f'Test accuracy of SVM with best hyperparameters {round(accuracy_score(y_test, y_test_pred),2)}')
```

Gaussian Naive Bayes:

```
## Naive Bayes Classifier
naive_bayes = GaussianNB()
train_scores = []
test_scores = []
for train_index, test_index in stratified_kfold.split(X, y):
    X_train_cv, X_test_cv = X.iloc[train_index], X.iloc[test_index]
    y_train_cv, y_test_cv = y.iloc[train_index], y.iloc[test_index]
    naive_bayes.fit(X_train_cv, y_train_cv)
train_score = naive_bayes.score(X_train_cv, y_train_cv)
train_scores.append(train_score)
test_score = naive_bayes.score(X_test_cv, y_test_cv)
test_scores.append(test_score)
print(f'Train accuracy of Gaussian Naive Bayes {round(np.mean(train_scores),2)}')
naive_bayes.fit(X_train, y_train)
y_train_pred = naive_bayes.predict(X_train)
y_test_pred = naive_bayes.predict(X_test)
print(f'Test accuracy of Gaussian Naive Bayes {round(np.mean(test_scores),2)}')
```

Multilayer perceptron classifier:

```
## MLP Classifier
mlp = MLPClassifier(max_iter=400, random_state=5805)
param_grid = {'hidden_layer_sizes': [(5, 5), (5, 10), (10, 10), (15, 15), (20, 20)]}
grid_search = GridSearchCV(mlp, param_grid, cv=stratified_kfold,scoring='accuracy')
grid_search.fit(X, y)
print("Best Hyperparameters:", grid_search.best_params_)

mlp = grid_search.best_estimator_
mlp.fit(X_train, y_train)
y_train_pred = mlp.predict(X_train)
y_test_pred = mlp.predict(X_test)
mlp_acc = accuracy_score(y_test, y_test_pred)
print(f'Train accuracy of Multilayer Perceptron with best hyperparameters {round(np.mean(mlp_acc),2)}')
print(f'Test accuracy of Multilayer Perceptron with best hyperparameters {round(accuracy_score(y_test, y_test_pred),2)}')
```

K-Means using silhouette analysis:

```
#K-Means Clustering
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
k_values = range(2, 11)
wcss = []
silhouette_scores = []
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=5085)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
    labels = kmeans.labels_
    silhouette_avg = silhouette_score(X, labels)
    silhouette_scores.append(silhouette_avg)
plt.figure(figsize=(8, 5))
plt.plot( *args: k_values, wcss, marker='o', linestyle='-')
plt.title("Elbow Method for Optimal k using KMeans Clustering")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Within-Cluster Sum of Squares (WCSS)")
plt.xticks(k_values)
plt.grid(True)
plt.show()
plt.figure(figsize=(8, 5))
plt.plot( *args: k_values, silhouette_scores, marker='o', linestyle='-')
plt.title("Silhouette Analysis for Optimal k using KMeans Clustering")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Silhouette Score")
plt.xticks(k_values)
plt.grid(True)
plt.show()
```

DBSCAN algorithm:

```
#Dbscan Algorithm
k = 57
nbrs = NearestNeighbors(n_neighbors=k).fit(X)
distances, indices = nbrs.kneighbors(X)
distances = np.sort(distances[:, -1])
plt.figure(figsize=(10, 6))
plt.plot(distances)
plt.title("K-Distance Graph")
plt.xlabel("Data Points (sorted)")
plt.ylabel(f"{k}-Distance")
plt.grid(True)
plt.show()
eps = 1.15
min_samples = 5
dbscan = DBSCAN(eps=eps, min_samples=min_samples)
clusters = dbscan.fit_predict(X)
df['Cluster'] = clusters
n_clusters = len(set(clusters)) - (1 if -1 in clusters else 0)
n_noise = list(clusters).count(-1)
print(f"Number of clusters: {n_clusters}")
print(f"Number of noise points: {n_noise}")
if n_clusters > 1:
    silhouette_avg = silhouette_score(X, clusters)
    print(f"Silhouette Score: {silhouette_avg}")
else:
    print("Silhouette Score is not defined for a single cluster.")
if isinstance(X, np.ndarray):
    X = pd.DataFrame(X, columns=[f"Feature_{i+1}" for i in range(X.shape[1])])
plt.figure(figsize=(10, 7))
if X.shape[1] >= 2:  # Ensure at least two features for plotting
```

Apriori algorithm:

```
df_le = df_apr[['DayOfWeek', 'Flight', 'Length', 'Time']]
df_le['DayOfWeek'] = pd.cut(df_le['DayOfWeek'], bins=3, labels=['StartOfWeek',
'MidOfWeek', 'Weekend'])
df_le['Flight'] = pd.cut(df_le['Flight'], bins=3, labels=['Small Aircraft', 'Medium Aircraft', 'Large Aircraft'])
df_le['Length'] = pd.cut(df_le['Length'], bins=3, labels=['Short Length', 'Average Length', 'Long Length'])
df_le['Time'] = pd.cut(df_le['Time'], bins=3, labels=['Midnight', 'Early Morning','Day'])
te = TransactionEncoder()
te_ary = te.fit(df_le.values).transform(df_le.values)
df_le = pd.DataFrame(te_ary, columns=te.columns_)
print(df_le.head(5))
df_le = df_le.astype('int')
print(df_le.head(5))
frequent_itemsets = apriori(df_le, min_support=0.1, use_colnames=True, verbose=1)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=0.2,num_itemsets=len(frequent_itemsets))
rules = rules.sort_values( by: ['confidence'], ascending=False)
print(rules.head(5).to_string())
```

References

1) Airlines Dataset to predict a delay. (n.d.). www.kaggle.com. https://www.kaggle.com/datasets/jimschacko/airlines-dataset-to-predict-a-delay
2) scikit-learn. (2019). scikit-learn: machine learning in Python. Scikit-Learn.org. https://scikit-learn.org/stable/
3) Raschka, S. (n.d.). mlxtend. Rasbt.github.io. https://rasbt.github.io/mlxtend/
4) Pandas. (2018). Python Data Analysis Library — pandas: Python Data Analysis Library. Pydata.org. https://pandas.pydata.org/
5) https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/
6) https://www.analyticsvidhya.com/blog/2021/06/understand-the-dbscan-clustering-algorithm/