IE 7275 Data Mining in Engineering

Flight Delay Prediction
Milestone: Final Report

Group 11
Aadarsh Siddha
Venkata Sai
Swathi
Gattamaneni

413-416-0390(Tel of Aadarsh)
617-516-7948 (Tel of Swathi)

siddha.a@northeastern.edu
gattamaneni.v@northeastern.edu

Percentage of Effort Contributed by Student1: 50%

Percentage of Effort Contributed by Student2: 50%

Signature of Student 1: Aadarsh Siddha

Signature of Student 2: Venkata Sai Swathi Gattamaneni

Submission Date: 5/21/2023

Problem Setting:

When a flight arrives after its stated arrival time, it is said to be delayed. Environmental factors are the main factor affecting this delay. Passengers find flight delays annoying, and airlines and nations suffer excruciatingly large financial losses as a result. An essential instrument that can assist aviation authorities in efficiently reducing aircraft delays is a structured prediction system. Air transportation delay indicates the lack of efficiency of the aviation system. It is a high cost to both airline companies and their passengers.
Therefore, predicting flight delays can improve airline operations and passenger satisfaction, which will result in a positive impact on the economy. Accurately forecasting flight delays can benefit both airlines and passengers by enabling them to better plan their travel and respond appropriately to delays. In order to study past flight data and uncover trends and insights that can be applied to the prediction of flight delays, data mining techniques can be used.
The Response Variable of flight delay project is both numeric prediction and classification.
The flight and weather data were combined into a single dataset and preprocessed to train a machine learning model that predicts the flight delay.

Problem Definition:

Using previous flight data, the problem of flight delay prediction in data mining entails creating a predictive model that can precisely estimate the likelihood of an aircraft being delayed. The objective is to examine previous flight data, find trends and insights, and create a predictive model that can provide reliable forecasts for impending flights. The predictive model can assist airlines and passengers in making travel plans, helping them respond appropriately to delays, and enhancing the effectiveness of air travel.

Key Components of the Problem:

1. The historical flight data used to train the prediction model typically consists of information about the airline, flight schedules, departure and arrival times, weather, and other pertinent characteristics. Many sources, including airline databases, meteorological databases, and publicly accessible datasets, can be used to gather this data.
2. Forecasting the probability that an aircraft will be delayed is the prediction task, which is often modeled as a binary classification issue. Based on the features

provided as input and trends discovered from previous data, the prediction model should be able to distinguish between delayed and un-delayed aircraft with accuracy.

3. Finding and choosing the most pertinent characteristics or variables that affect flight delays is known as feature engineering. This may involve elements like the airline, the origin and destination airports, the departure and arrival times, the time of day, the weather, and other pertinent variables. In order to ensure that the data is in an appropriate format for model training, feature engineering may also comprise data pretreatment, standardization, and transformation.
4. Model Development: A variety of data mining techniques, including machine learning algorithms, statistical techniques, and other methodologies, can be used to create the prediction model. The precise requirements of the task and the qualities of the data determine which model should be used.
5. Model Evaluation: The accuracy, precision, recall, F1-score, and ROC curve are some relevant assessment metrics that should be used to assess the performance of the predictive model. To evaluate the model's generalization performance and guarantee its trustworthiness in producing precise predictions, it should be verified on a different test dataset.
6. Interpretability: Understanding the variables that affect flight delays and gaining new insights rely on the predictive model's interpretability. An interpretable model can assist in making decisions and taking appropriate action to alleviate flight delays by assisting in the identification of the most important aspects and in comprehending the underlying patterns.

Data Sources:

We Collected two datasets from GitHub repository which is of flight delay, and which is of weather data and then we merged both data set into one data.

Flight Data:

https://drive.google.com/drive/folders/1SG-U-9j-kq79JT3_M3jowiZBjUTQLaqf

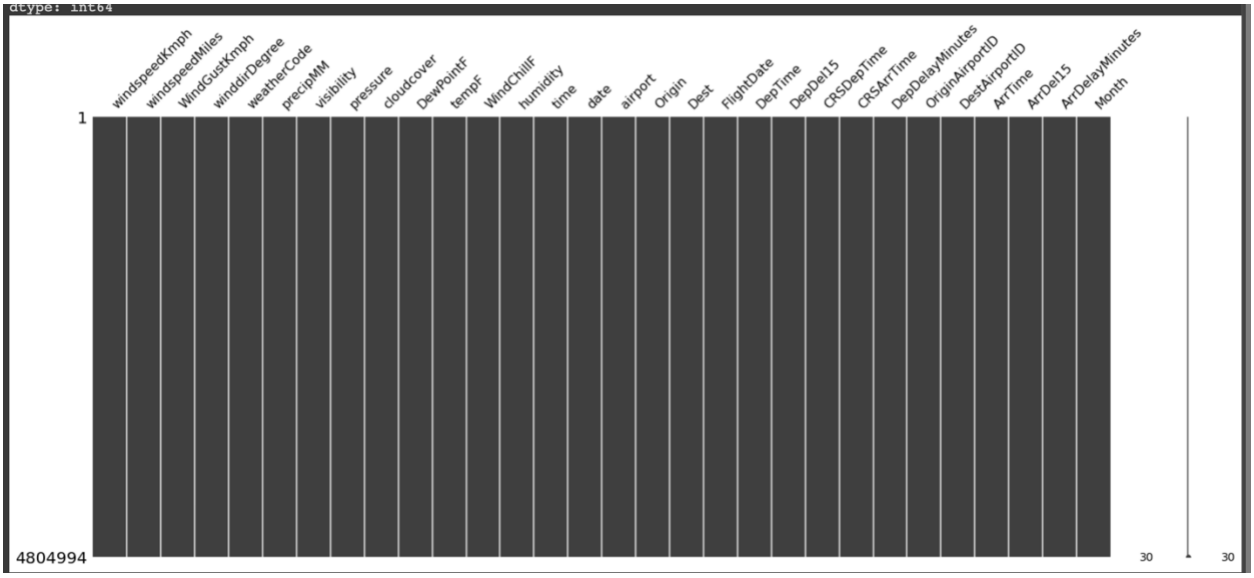
weather Data:

https://drive.google.com/drive/folders/1FH3SzcDlcDVy4QkwB7z4VNi1bE18d0JA

Data Description:

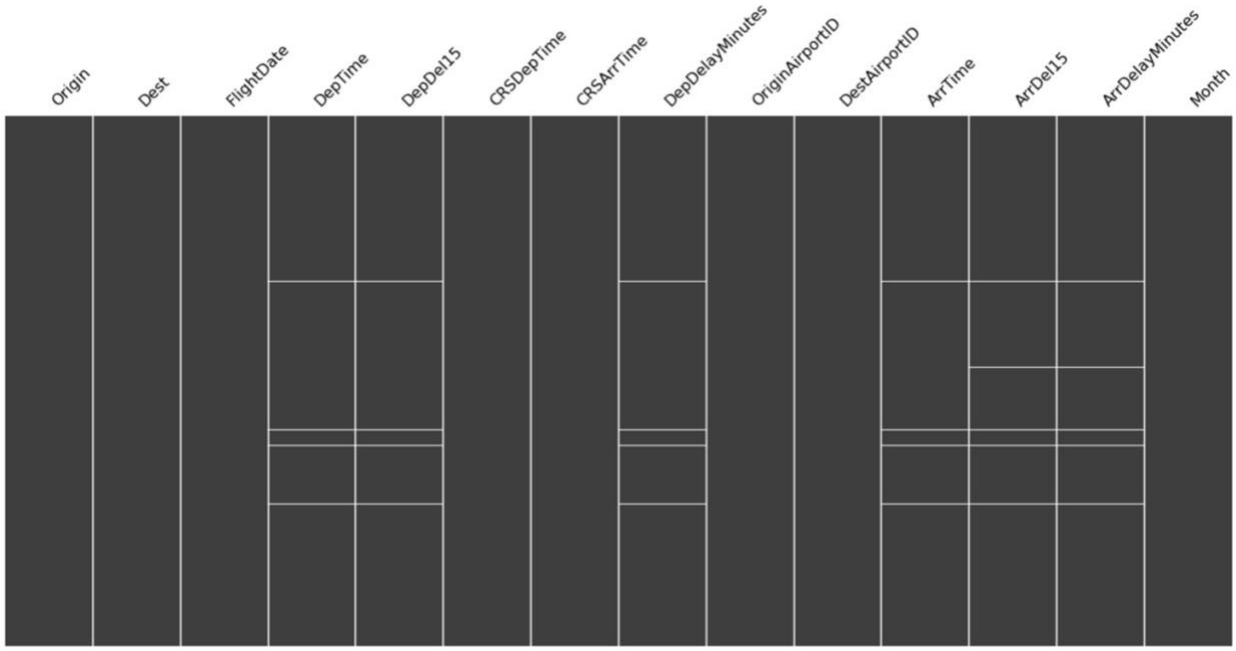
Our Dataset contains 1,12,92,279 rows and 110 columns. After cleaning the dataset, it has 11292279 rows and 30 columns which we have decided as a features. Typically, previous flight data such as flight schedules, departure and arrival times, weather conditions, airline information, and other pertinent variables are utilized to estimate flight delays. Finally, we chose 16 features for our classifier.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11292279 entries, 42830 to 7194062
Data columns (total 14 columns):
 #   Column           Dtype
---  ------           -----
 0   Origin           object
 1   Dest             object
 2   FlightDate       datetime64[ns]
 3   DepTime          float64
 4   DepDel15         float64
 5   CRSDepTime       int64
 6   CRSArrTime       int64
 7   DepDelayMinutes  float64
 8   OriginAirportID  int64
 9   DestAirportID    int64
 10  ArrTime          float64
 11  ArrDel15         float64
 12  ArrDelayMinutes  float64
 13  Month            int64
dtypes: datetime64[ns](1), float64(6), int64(5), object(2)
memory usage: 1.3+ GB
```



Data Exploration:

The white lines that show in the below figure indicates the missing values in our data set.



These are the variables that we dropped due to the below mentioned reason.

Columns to be Eliminated:

Feature	Reason to Eliminate
WindSpeedMiles, WindGustKmph	High correlation to WindSpeedKmph
CRSArrTime, ArrTime, ArrDelay15	Directly Related to Target Variable
WeatherCode,precipMM,visibility,cloudcover,humidity	Reduced by PCA
DewPointF,WindChillF,tempF	Reduced by PCA
OriginAirportID,DestAirportID	Same as Origin,Dest
airport	Same as Origin
FlightDate,date	Extracted as Month

Target variables was highly correlated with a few variables, those variables were eliminated. Most of our data was related to weather conditions, therefore it was prone to outliers but couldn't be removed. Variables such windspeed, temp and humidity have normal distributions We had around 90,000 null values initially in the flight dataset, all these values were dropped. Correlation heatmaps helped us identify correlated values and reduce using PCA. We Used Hist plot, boxplot and scatter plot to our columns.

The below mentioned figure indicates the detail description of our data like

```
df_flight["pressure"].describe()

count    4.804994e+06
mean     1.016223e+03
std      6.401458e+00
min      9.840000e+02
25%      1.012000e+03
50%      1.016000e+03
75%      1.020000e+03
max      1.046000e+03
Name: pressure, dtype: float64
```

```
df_flight["cloudcover"].describe()
```

```
count    4.804994e+06
mean     4.198709e+01
std      3.964129e+01
min      0.000000e+00
25%      0.000000e+00
50%      3.300000e+01
75%      8.300000e+01
max      1.000000e+02
Name: cloudcover, dtype: float64
```

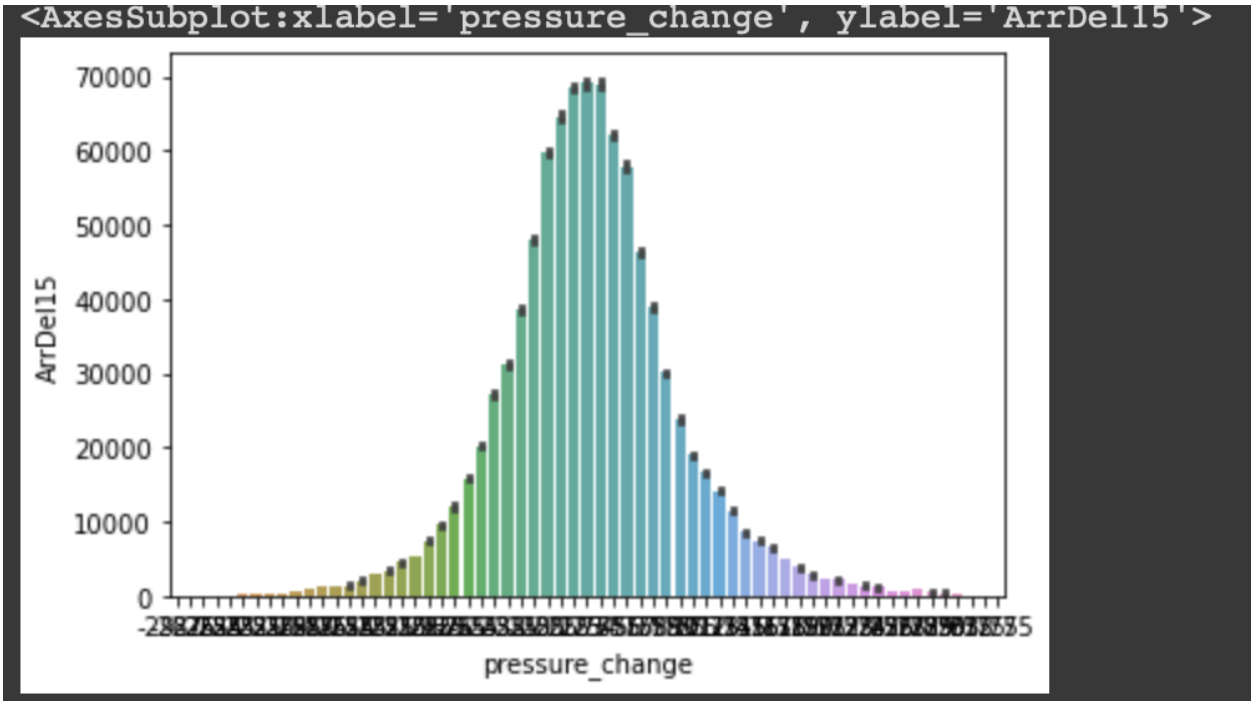
```
[ ] df_flight["visibility"].describe()
```

```
count    4.804994e+06
mean     9.457986e+00
std      1.778129e+00
min      0.000000e+00
25%      1.000000e+01
50%      1.000000e+01
75%      1.000000e+01
max      2.000000e+01
Name: visibility, dtype: float64
```

```
[ ] df_flight["humidity"].describe()
```

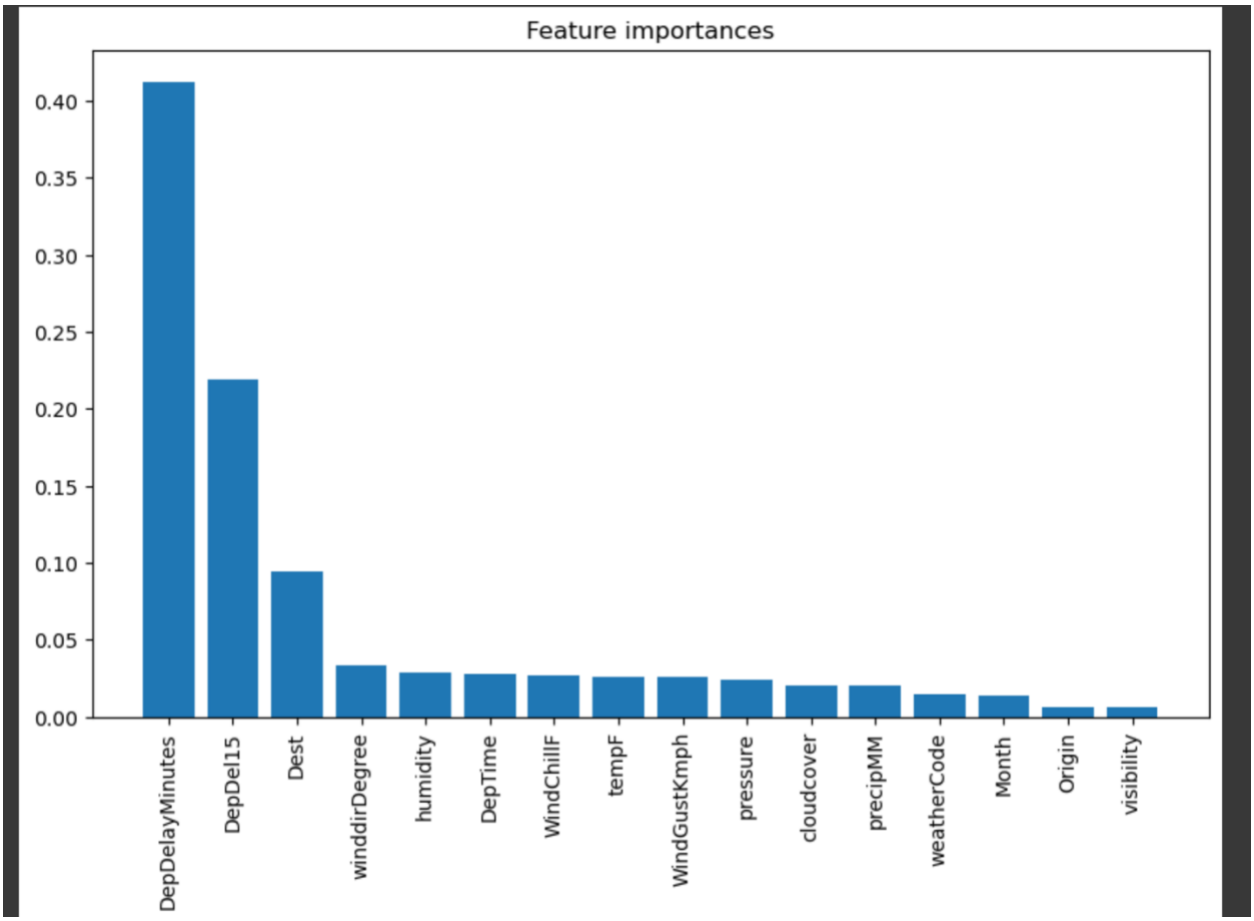
```
count    4.804994e+06
mean     6.211080e+01
std      2.229466e+01
min      3.000000e+00
25%      4.700000e+01
50%      6.500000e+01
75%      8.000000e+01
max      1.000000e+02
Name: humidity, dtype: float64
```

ArrDel15 is the one of the features to our dataset. This barplot is used to estimate the pressure_chance of ArrDel15.



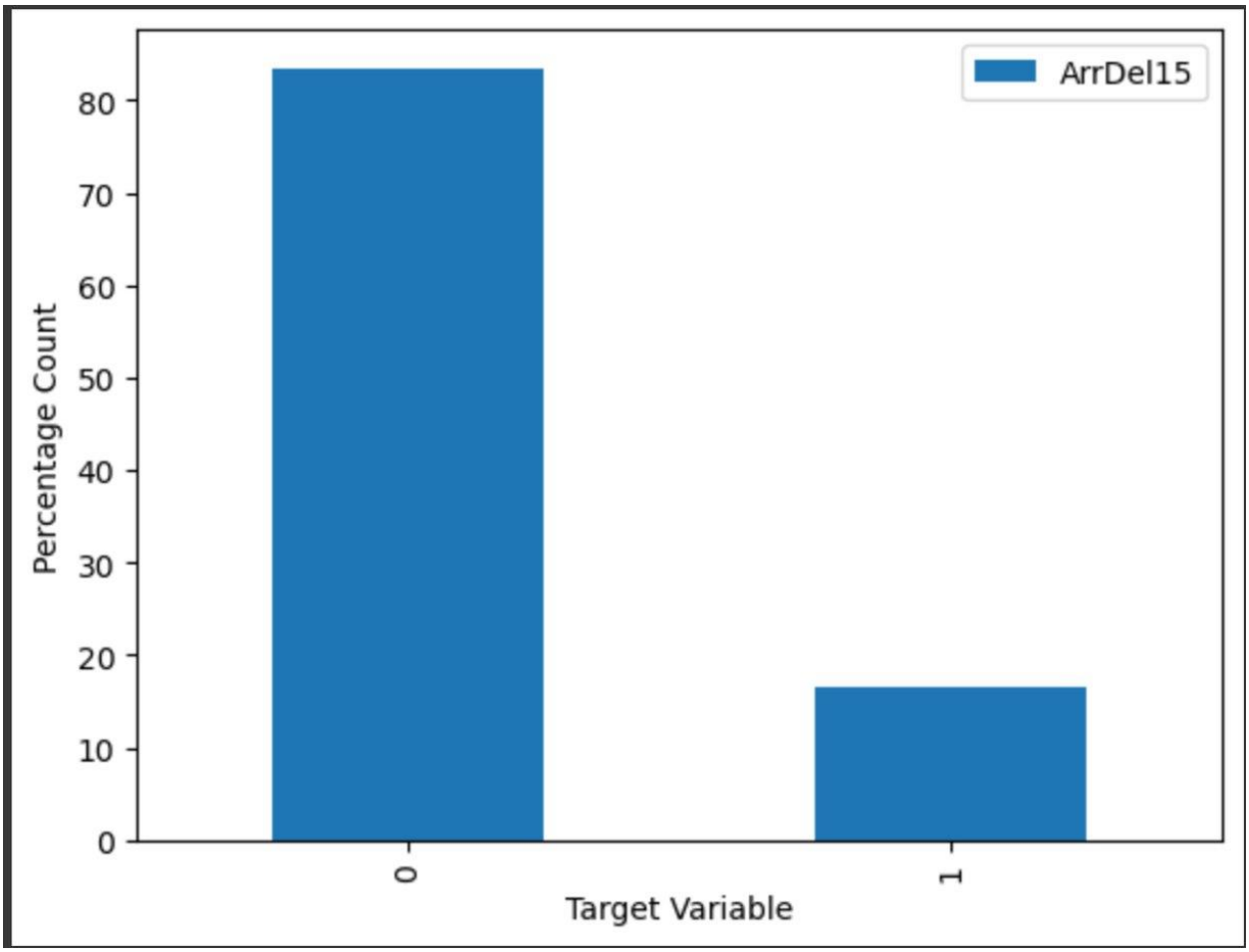
Feature Importance:

This is to get an idea on features. Which feature is more important and how important it is for the model. Based on this plot we chose DepDelayMinutes and Arrdel15 as our two target variables.



Data Mining Task:

We had an imbalance percentage in target variable which is almost 85percentage to 0 and 15 percentage to 1



PCA Reduction:

WeatherCode, PrecipMM, visibility, cloudcover, humidity are the five features to the model which is highly correlated to each other and gives the inaccuracy measures. By using PCA reduction we converted these five features to three PCA components which has o correlation to other which says the same meaning to our model.

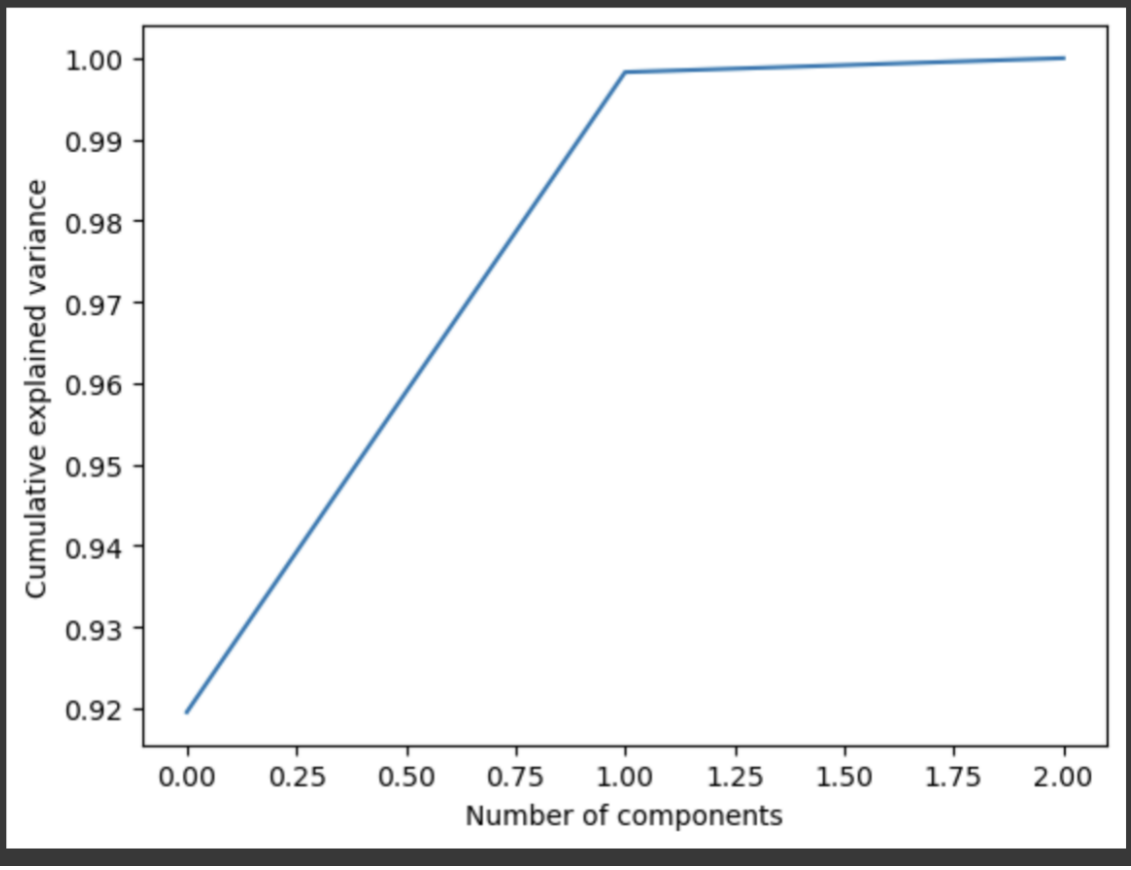
	weatherCode	precipMM	visibility	cloudcover	humidity
weatherCode	1.00	0.53	-0.35	0.48	0.29
precipMM	0.53	1.00	-0.32	0.31	0.27
visibility	-0.35	-0.32	1.00	-0.37	-0.41
cloudcover	0.48	0.31	-0.37	1.00	0.45
humidity	0.29	0.27	-0.41	0.45	1.00

	Component1	Component2	Component3
Component1	1.00	0.00	-0.00
Component2	0.00	1.00	0.00
Component3	-0.00	0.00	1.00

PCA object to fit into our model:

```
# Step 1: Create a PCA object and fit it to the data
pca = PCA().fit(df2)

# Step 2: Plot the scree plot
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of components')
plt.ylabel('Cumulative explained variance')
plt.show()
```



Models Implemented:

We Implemented Four Regressors and Five Classifier which are

Regressor
Linear Regression
Random Forest Regressor
KNN Regressor
Neural Net

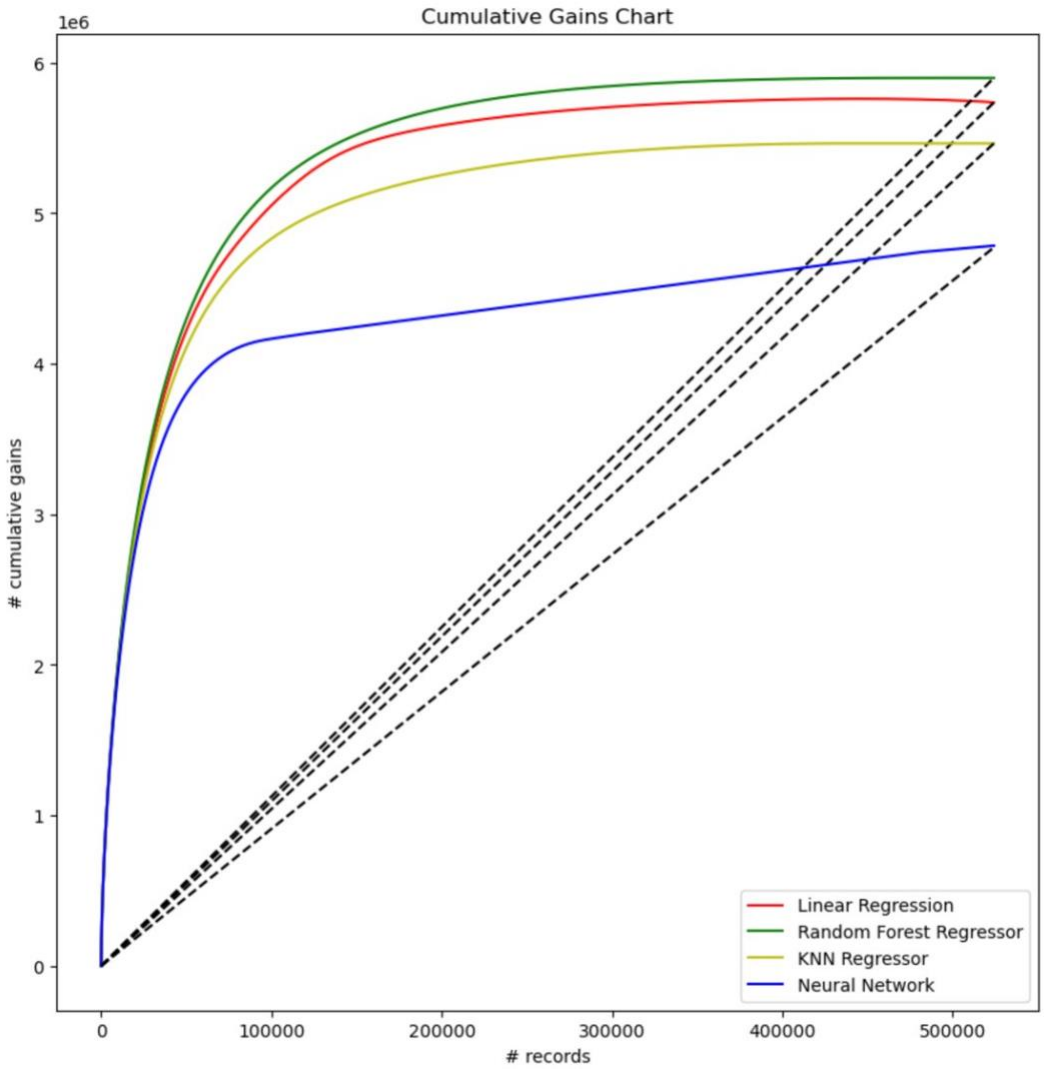
Classifier
Logistic Regression
Decision Tree Classifier
Random Forest Classifier
KNN
Neural Net
Linear Discriminant Analysis

Accuracy Metrics for Regressor:

Model_Name	R2-Score	MSE	RMSE	MAE	Pearson-R	p-value
Linear Regression	0.946566	71.720215	8.468779	4.009742	0.972917	0.0
RandomForestRegressor	0.948619	68.965437	8.304543	4.023837	0.974044	0.0
KNN-Regressor	0.936540	85.178228	9.229205	4.303169	0.968030	0.0
Neural Network	0.936213	85.617085	9.252950	4.329141	0.969166	0.0

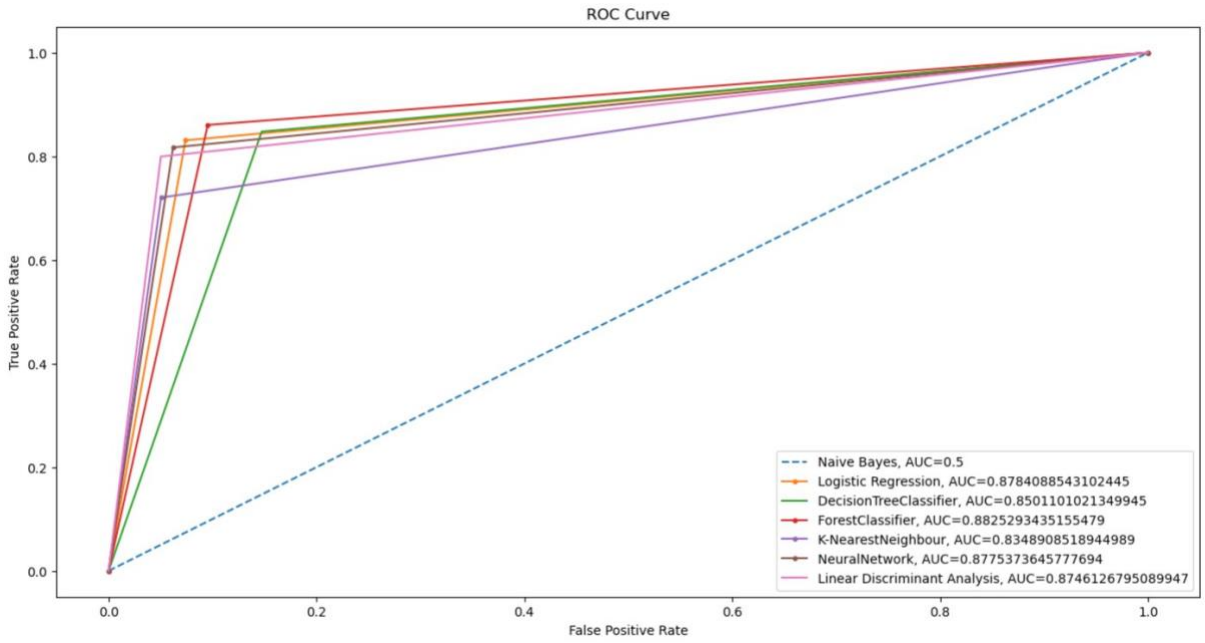
Gains Chart for Regression:

This is the gains chart for regression



ROC Cures for Classifer:

- Navie Bayes, AUC = 0.5
- Logistic Regression, AUC – 0.8784
- Decision Tree Classifier, AUC = 0.8501
- ForestCLassifier, AUC=0.8825
- K-NearestNeighbour, AUC = 0.8348
- NeauralNetwork, AUC=0.8775
- Linear Discriminant Analysis, AUC = 0.8746



Accuracy Metrics for Classifier:

Logistic regression:

	Method	Accuracy	Class_0_Precision	Class_0_Recall	Class_0_F1_score	Class_1_Precision	Class_1_Recall	Class_1_F1_score
0	Logistic-Regression	0.878222	0.844488	0.926005	0.88337	0.918809	0.830813	0.872598

Decision Tree classifier:

	Method	Accuracy	Class_0_Precision	Class_0_Recall	Class_0_F1_score	Class_1_Precision	Class_1_Recall	Class_1_F1_score
0	DecisionTreeClassifier	0.8501	0.847328	0.852647	0.849979	0.852885	0.847573	0.850221

Random Forests:

	Method	Accuracy	Class_0_Precision	Class_0_Recall	Class_0_F1_score	Class_1_Precision	Class_1_Recall	Class_1_F1_score
0	Random-Forest-Classifier	0.8501	0.847328	0.852647	0.849979	0.852885	0.847573	0.850221

KNN:

	Method	Accuracy	Class_0_Precision	Class_0_Recall	Class_0_F1_score	Class_1_Precision	Class_1_Recall	Class_1_F1_score
0	KNN	0.83444	0.771072	0.949465	0.85102	0.934923	0.720317	0.813708

Neural Net:

	Method	Accuracy	Class_0_Precision	Class_0_Recall	Class_0_F1_score	Class_1_Precision	Class_1_Recall	Class_1_F1_score
0	Neural-Net	0.874404	0.822328	0.95392	0.883249	0.945652	0.795511	0.864108

	Method	Accuracy	Class_0_Precision	Class_0_Recall	Class_0_F1_score	Class_1_Precision	Class_1_Recall	Class_1_F1_score
0	Logistic-Regression	0.878222	0.844488	0.926005	0.883370	0.918809	0.830813	0.872598
0	DecisionTreeClassifier	0.850100	0.847328	0.852647	0.849979	0.852885	0.847573	0.850221
0	Random-Forest-Classifier	0.850100	0.847328	0.852647	0.849979	0.852885	0.847573	0.850221
0	KNN	0.834440	0.771072	0.949465	0.851020	0.934923	0.720317	0.813708
0	Neural-Net	0.876450	0.851727	0.910414	0.880093	0.904593	0.842752	0.872578
0	LDA	0.874317	0.824584	0.949667	0.882717	0.941214	0.799558	0.864622

Performance Evaluation:

Linear Regression Model Metrics

```
from sklearn.linear_model import LinearRegression

# create a linear regression model and fit the data
model = LinearRegression()
model.fit(X_train,Y_train)


from sklearn.metrics import r2_score, mean_squared_error

# Predict the target values for the test set
linear_reg_pred = model.predict(X_test)
```

```
# Calculate the metrics
r2 = r2_score(Y_test, linear_reg_pred)
mse = mean_squared_error(Y_test, linear_reg_pred)
rmse = np.sqrt(mse)

# Print the metrics
print('R2 score:', r2)
print('MSE:', mse)
print('RMSE:', rmse)
```

```
R2 score: 0.9469113010676172
MSE: 71.34788292526686
RMSE: 8.446767602181728
```

Random Forest Regressor:

```
from sklearn.ensemble import RandomForestRegressor


# create a Random Forest Regressor object with 50 trees
rf_regressor = RandomForestRegressor(n_estimators=50, random_state=42)

# fit the training data to the model
rf_regressor.fit(X_train, Y_train)

# predict the target values for the test set
rf_pred = rf_regressor.predict(X_test)
```

```
/var/folders/77/l0m4syw9301bj1fcnkqk75fr0000gn/T/ipykernel_76670/20695096
  rf_regressor.fit(X_train, Y_train)
```

```
[ ] # calculate the metrics
    mse = mean_squared_error(Y_test, rf_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(Y_test, rf_pred)

    # print the metrics
    print("MSE:", mse)
    print("RMSE:", rmse)
    print("R2 Score:", r2)
```

```
MSE: 67.65235783069227
RMSE: 8.225105338577267
R2 Score: 0.9496610759887377
```

KNN Regressor:

```
[ ]  from sklearn.neighbors import KNeighborsRegressor

     # create a KNN Regressor
     knn_regressor = KNeighborsRegressor(n_neighbors=10)

     # fit the training data to the model
     knn_regressor.fit(X_train, Y_train)

     # predict the target values for the test set
     KNN_pred = knn_regressor.predict(X_test)
```

```
     # calculate the metrics
     mse = mean_squared_error(Y_test, KNN_pred)
     rmse = np.sqrt(mse)
     r2 = r2_score(Y_test, KNN_pred)

     # print the metrics
     print("MSE:", mse)
     print("RMSE:", rmse)
     print("R2 Score:", r2)
```

```
MSE: 81.84810346088189
RMSE: 9.046994167174084
R2 Score: 0.9390982725111459
```

Neural Net Regressor:

```
# calculate the metrics
mse = mean_squared_error(Y_test, NN_pred)
rmse = np.sqrt(mse)
r2 = r2_score(Y_test, NN_pred)

# print the metrics
print("MSE:", mse)
print("RMSE:", rmse)
print("R2 Score:", r2)
```

```
MSE: 70.90687718583733
RMSE: 8.420622137694894
R2 Score: 0.9472394456455377
```

Project Results and Impact of the Project Outcomes:

- For Classifier Model Random Forest Classifier is the best model for our dataset. The AUC for Random Forest comes out to be 0.88. Other metrics for random forest are as follows:

Method	Accuracy	Class_0_Precision	Class_0_Recall	Class_0_F1_score	Class_1_Precision	Class_1_Recall	Class_1_F1_score
Random-Forest-Classifier	0.8501	0.847328	0.852647	0.849979	0.852885	0.847573	0.850221

- It is 85% accurate in predicting our target class 1

- For Regression model Random Forest Regressor is the best model for our dataset. The model shows maximum gains in minimum number of records.

Model_Name	R2-Score	MSE	RMSE	MAE	Pearson-R	p-value
RandomForestRegressor	0.948619	68.965437	8.304543	4.023837	0.974044	0.0

- Its R2 Score is 0.94 and gives an error of 8 mins in predicting exact flight delay in minutes.
- Our model can be improved collecting more weather data for the airports in USA, resulting in delay prediction for all airports.
- Also, we need to develop a pipeline to connect both models and resulting in improved decision making and better accuracy for flight delay.
- Our model can be used to develop apps, predicting chance of flight delay in advance because passengers get informed about flight delays only hours before their journey and can save time.