Flight Delay Prediction

Introduction to Machine Learning,
Spring 2023

Team:

S.No	Name
1.	Shivakumar Suresh
2.	Sinchana Sulugodu Shashidhara
3.	Srinidhi Shivaram Prakash
4.	Jahnavi Pathapati

Contents:

1. Abstract and Problem Statement:

There is a great expansion of the aviation industries in the present world. This has also introduced an increased delay in flights due to the air-traffic jams. Around 20% of airline flights are cancelled or delayed annually, costing travellers more than $20 billion in lost time and money. Passengers, airlines, and airports suffer considerable financial and other losses due to flight delays. As a result, anticipating the possibility of a delay becomes an important task.

The goal of this project is to predict flight delays, which are the biggest economic drag on many nations and the fastest and most comfortable mode of transportation. By using machine learning algorithms, we are trying to identify the status of flights to an airport which can in-turn explain reasons for delays, saving huge amounts of turnovers.

2. Introduction:

One of the major business problems that airlines encounter is the cost of flights being delayed due to operational failures and natural disasters. This is a costly affair for the airlines and causes problems for customers with scheduling and operations, damaging their reputation and losing them customers. As airline company customers, it is common knowledge that neither we nor the airline business's ground staff typically receives advance notice of flight delays due to a variety of factors. However, it is known that weather delays flights more frequently than any other factor. This encourages us to compute the delay on the wing prior to departure using the current weather information along with multiple factors.

Chicago (ORD), Denver (DEN), Newark (EWR), and Washington (IAD) are the four locations from which United Airlines offers direct flights to Syracuse (SYR). The objective of our project is to forecast, 1-4 days in advance, if each flight will arrive into SYR early, on schedule, late, or extremely late. The new column called “Status” had to be predicted into the csv file.

The dataset was collected from the Bureau of Transportation Statistics. Even though the information we gathered was very limited, it helped us understand how the weather affects flight delays. The purpose of this research is to develop machine learning models and employ Exploratory Data Analysis to forecast arrival delays. The column “Arrival Delay” in the csv file had to be correlated with each feature and visualised. Heatmaps and other graphs were used to check the relationship. The algorithm's main application will be in the prediction of potential delays for the United Airlines flights coming into Syracuse airport on specific days.

3. Data Collection:

3.1 Flight Data:

The flight data that we took is from the Bureau of Transportation Statistics for the year 2021 to 2023. The dataset consists of 1364 data points with 17 columns, including 7 categorical variables and 10 numerical variables

Feature	Data Type
Carrier Code	Categorical (String - Two letter code: UA)
Date	Categorical (MM/DD/YYYY format)
Flight Number	Numerical
Tail Number	Categorical
Origin Airport	Categorical (String - Three letter identifier)
Scheduled Arrival Time	Categorical (hh:mm format)
Actual Arrival Time	Categorical (hh:mm format)
Scheduled Elapsed Time	Numerical
Actual Elapsed Time	Numerical
Actual Delay	Numerical
Wheels-on-time	Categorical (hh:mm format)
Taxi-In time	Numerical
Delay Carrier	Numerical
Delay Weather	Numerical
Delay National Aviation System	Numerical
Delay Security	Numerical
Delay Late Aircraft Arrival	Numerical

3.2 Weather Data:

We extracted the weather data for Syracuse and origin airport, from the weatherbit website. The dataset consists of 364 data points with 35 columns, including 2 categorical variables and 33 numerical variables.

For predictions we collected the forecast data.
Weather History Data collection : **https://api.weatherbit.io/v2.0/history/daily**
Weather Forecast Data collection: **https://api.weatherbit.io/v2.0/forecast/daily**

Weather Forecast data is used for predictions.

Feature	**Data Type**
Clouds	Numerical
datetime	Categorical (YYYY-MM-DD format)
dewpt	Numerical
dhi	Numerical
dni	Numerical
ghi	Numerical
max_dhi	Numerical
max_dni	Numerical
max_ghi	Numerical
max_temp	Numerical
max_temp_ts	Numerical
max_uv	Numerical
max_wind_dir	Numerical
max_wind_spd	Numerical
max_wind_spd_ts	Numerical
min_temp	Numerical
min_temp_ts	Numerical

precip	Numerical
pprecip_gpm	Numerical
pres	Numerical
revision_status	Categorical
rh	Numerical
slp	Numerical
snow	Numerical
snow_depth	Numerical
solar_rad	Numerical
t_dhi	Numerical
t_dni	Numerical
t_ghi	Numerical
t_solar_rad	Numerical
temp	Numerical
ts	Numerical
wind_dir	Numerical
wind_gust_spd	Numerical
wind_spd	Numerical

4 Exploratory Data Analysis:

We have to analyse the data in order to find out what feature has the highest correlation to understand arrival delay. Even though the dataset contained 1364 data points with 17 columns, including 7 categorical variables and 10 numerical variables, not all of them were timely. For instance, the flight number hardly ever affects delays. Additionally, because features like taxi wait time and wheels-off time are directly related to departure time, these features shouldn't have any bearing on our data set. We chose a number of variables for the data analysis that may have some correlations with our goal and could improve our forecasts.

The categorical variables were Carrier Code, Date, Tail Number, Origin Airport, Scheduled Arrival Time, Actual Arrival Time, and Wheels-on Time. The numerical variables are Flight_Number, Scheduled Elapsed Time (Minutes), Actual Elapsed Time (Minutes), and Arrival Delay (Minutes). To perform our analysis, we split the data into an 80-20 ratio for training and testing purposes. During the exploratory data analysis (EDA), we found that Arrival Delay is correlated with Scheduled Elapsed Time (Minutes), Actual Elapsed Time (Minutes), Delay Late Aircraft Arrival (Minutes), Delay Carrier (Minutes), and Delay Weather (Minutes). We used heatmaps and plotted graphs to visualise the correlations between Arrival Delay and each feature.

Based on the results of our EDA, few columns had to be removed and new ones had to be added to increase correlations.

Features considered for data analysis:

Feature	DataType	Significance
Date	datetime64(ns)	Depending on the date, factors like weather and holidays could affect the arrival time of flights
Day	int64	Weekends might have more air traffic compared to weekdays hence it was created using “date” column and ‘strfitime’ function
Origin Airport	object	Direct flights coming into Syracuse
Flight Number	object	To ensure we are considering United Airlines flights only hence merged carrier code with flight_number
Arrival Time	object	To check if there has been a delay
Scheduled Elapsed Time (Minutes)	int64	Time that is intended for a given flight
month	int64	Month as an integer

org_clouds	int64	average cloud coverage at origin airport
org_pres	float64	Pressure at origin airport affecting the flight arrival
org_snow	float64	Inches of snow at origin airport affecting the flight arrival
org_temp	float64	Temperature at origin airport affecting the flight arrival
org_wind_spd	float64	Wind speed at origin airport affecting the flight arrival
dest_clouds	int64	average cloud coverage at destination airport
dest_pres	float64	Pressure at destination airport affecting the flight arrival
dest_snow	int64	Inches of snow at destination airport affecting the flight arrival
dest_temp	float64	Temperature at destination airport affecting the flight arrival
dest_wind_spd	float64	Wind speed at destination airport affecting the flight arrival
Flight_Number	object	Integer part of the Airline number
Scheduled Arrival Time	float64	Actual arrival time used to check for delays

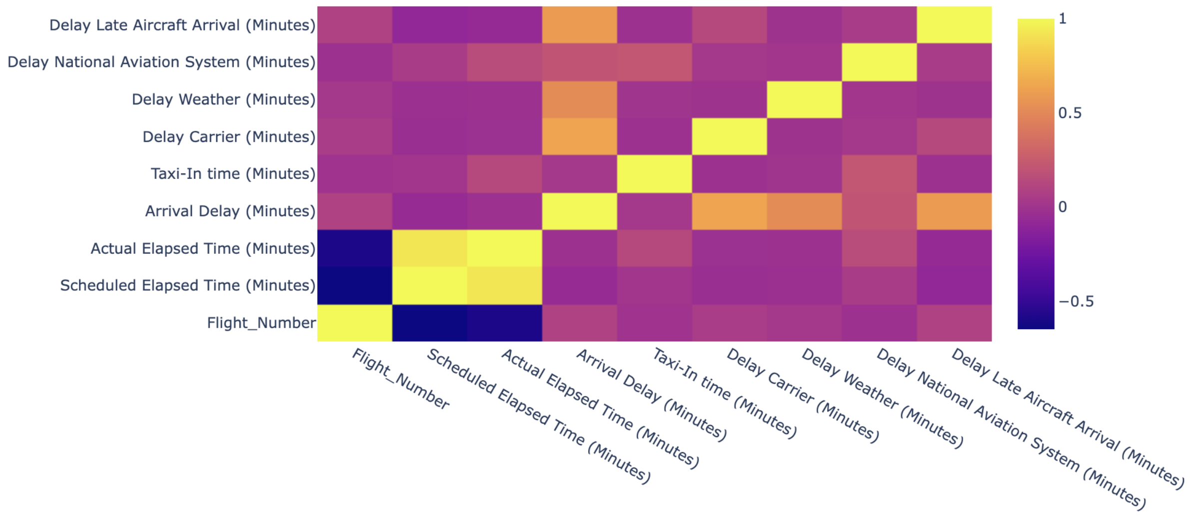

Figure 1: Heat Map for the flight data

The features "Actual Elapsed Time (Minutes)" and "Scheduled Elapsed Time (Minutes)" have a strong positive correlation with a coefficient of 0.92, according to the heatmap.
A correlation coefficient of 0.92 indicates a strong positive linear relationship between these two features. As a result, flights with longer scheduled durations frequently take longer than expected to complete, and vice versa. Due to their strong correlation, it is possible to predict a flight's actual elapsed time from its scheduled elapsed time.

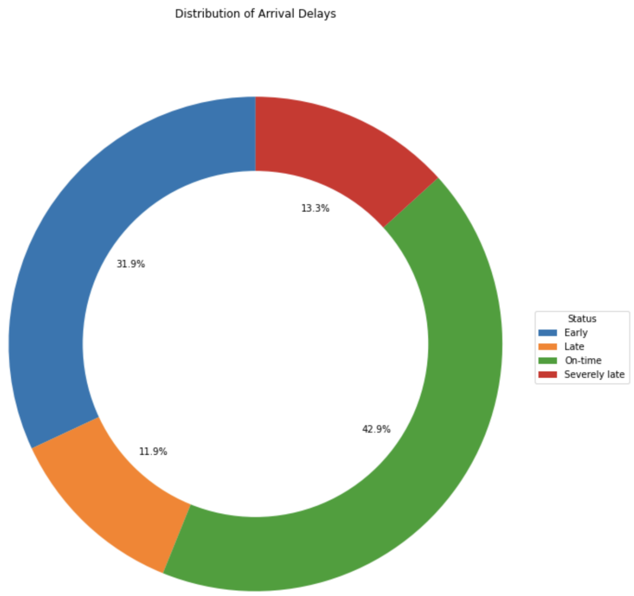

Figure 2: Distribution of Arrival Delays

The pie chart presents the distribution of arrival delays based on the status of flights. It can be helpful in identifying which status of flights have the highest percentage of delays, which is be useful for the initial predictions

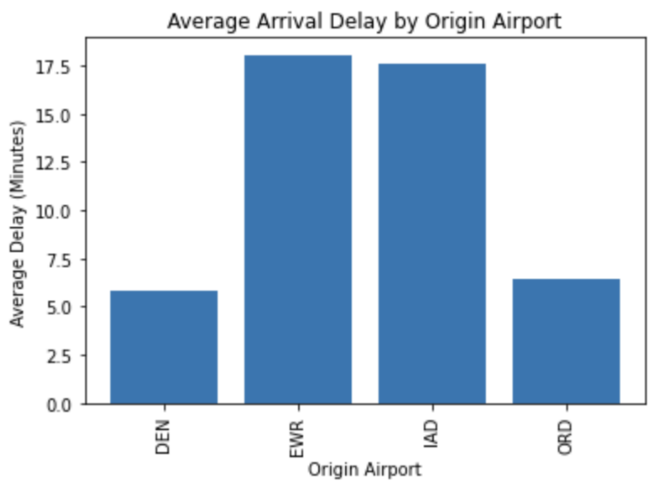

Figure 3: Average Arrival Delay by Origin Airport

The bar chart where each bar represents the average delay for each origin airport. While it does not directly help in flight status prediction, it is useful for understanding the performance of different airports.

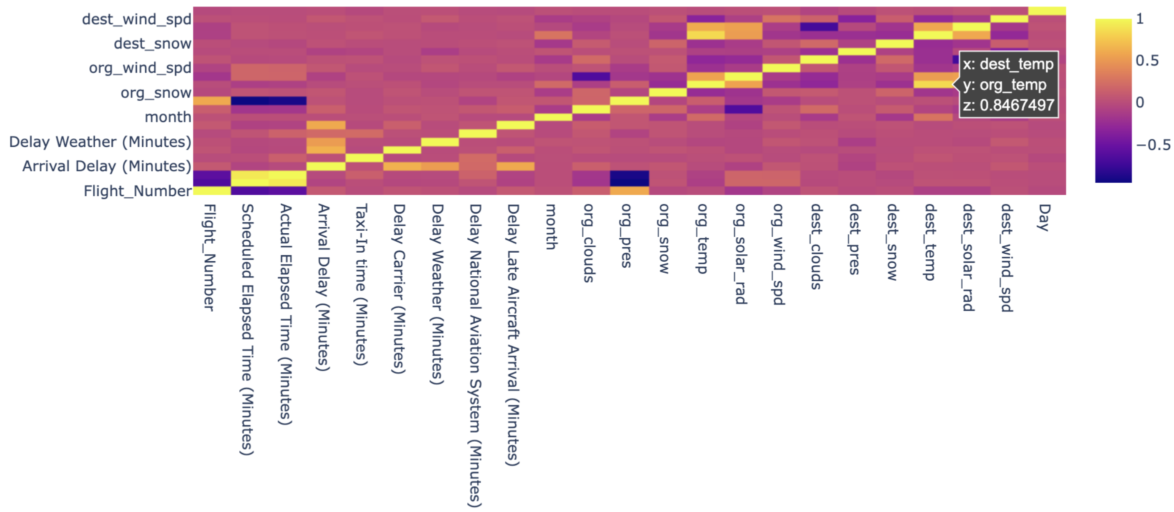

Figure 4: Heat Map for flight and weather data

Based on the heatmap generated from the given code, we can observe that there is a strong positive correlation of 0.84 between the 'dest_temp' and 'org_temp' features.

5. Data Pre-processing and Feature selection:

5.1. Flight Data used:

The Flight data which are known in advances are selected, they are:

a) The target variable '*Arrival Delay (Minutes)*' is converted into a categorical variable named 'Status' having four categories.
Below is the code snippet:

```
def classify_status(delay):
    if delay <= -10:
        return 'Early'
    elif delay >= -10 and delay <= 10:
        return 'On-time'
    elif delay > 10 and delay <= 30:
        return 'Late'
    else:
        return 'Severely late'
```

b) The date column is we extracted month and day of the week:
   1. *Month:* Integer value (ordinal),it can take any discrete value in the range (1-12)
   2. Day: Integre value (ordinal), it can take any discrete value in the range (0-6)

c) *Flight_Number*: Categorical variable, represents the flight number of the given airline.

d) *Origin Airport*: Categorical variable, represents the origin Airport of the flight

e) Scheduled Arrival Time: Float, represents the scheduled arrival time at Syracuse airport, in 24Hrs format, minutes are converted into fraction of hours.

f) Scheduled Elapsed Time: Float, represents the estimated duration of travel.

5.2. Weather data used:

The weather data which can affect the fight is being considered, they are:

a) *org_clouds:* int, represents the clouds at the origin airport

b) *org_pres:* float, represents the atmospheric pressure at the origin airport

c) *org_snow:* float, represents the snow level at the origin airport

d) *org_temp:* float, represents the temperature at the origin airport

e) *org_wind_spd:* float, represents the wind speed at the origin airport

f) *dest_clouds:* int, represents the clouds at the syracuse airport

g) *dest_pres:* float, represents the atmospheric pressure at the Syracuse airport

h) *dest_snow:* int, represents the snow level at the Syracuse airport

i) *dest_temp:* float, represents the temperature at the Syracuse airport

j) *dest_wind_spd:* float, represents the wind speed at the Syracuse airport

Pre-processed data:

Total 15 predictors and one target variable:

```
Data columns (total 17 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   Flight_Number                      1361 non-null   object
 1   Origin Airport                     1361 non-null   object
 2   Scheduled Arrival Time             1361 non-null   float64
 3   Scheduled Elapsed Time (Minutes)   1361 non-null   int64
 4   month                              1361 non-null   int64
 5   Day                                1361 non-null   int64
 6   org_clouds                         1361 non-null   float64
 7   org_pres                           1361 non-null   float64
 8   org_snow                           1361 non-null   float64
 9   org_temp                           1361 non-null   float64
 10  org_wind_spd                       1361 non-null   float64
 11  dest_clouds                        1361 non-null   float64
 12  dest_pres                          1361 non-null   float64
 13  dest_snow                          1361 non-null   float64
 14  dest_temp                          1361 non-null   float64
 15  dest_wind_spd                      1361 non-null   float64
 16  Status                             1361 non-null   object
dtypes: float64(11), int64(3), object(3)
```

Note: The numeric values are scaled before training, testing and predictions.

6. Methods and Approaches:

6.1 Algorithms:

We choose to use built-in sklearn kit functions to build the prediction model. The types of algorithms we utilised, together with their benefits and drawbacks in relation to our dataset, are displayed in the following table:

Algorithm	Pros	Cons
Multinomial Logistic Regression	Analysis of different dependent variables can be done since it is flexible, yields very close to accurate predictions and it is very easy to understand these predictions	It assumes linear relationship between dependent variables, requires bigger sample size for better results, has a possibility of underfitting
Random Forest	Can produce accurate results even for larger datasets, has reduced variance and can be used in classification and regression problems	Complex to interpret visually, has a problem of overfitting and can be difficult to implement due to its complexity
Gradient Boosting	Produces high accuracy, it can also handle mixed and missing data, it provides insight to the importance of the features.	It can be a black-box model and is sensitive to overfitting
K-Nearest Neighbour	No assumptions hence easy to understand and implement, can be used for both classification and regression, easily implemented to multi-class problems	It can be slow with large datasets and can be sensitive to outliers

We have tried testing the above four algorithms for our data with various parameters, and found that 'Random Forest' performed better, with better F-1 score and accuracy, so we used it for the final predictions. The results of each mosel is discussed in the next section.

7. Results and Analysis:

Aircraft delay prediction is a common problem in the aviation industry, where we need to predict the likelihood of a flight being delayed based on various factors such as weather conditions, air traffic congestion, mechanical issues, etc. We used machine learning models to analyse historical flight data and identify patterns that could help predict flight delays.

In our aircraft delay algorithm, we split the airlines delayed data into a 0.09 test size. This means that 123 of the 1361 total data points were used for testing, while the remaining 1238 data points were used for training.

We evaluated the performance of the three prime machine learning models (excluding preliminary Logistic Regression) using various evaluation metrics such as precision, recall, and F1 score. Precision measures the proportion of true positives out of all the positive predictions made by the model, while recall measures the proportion of true positives out of all the actual positive instances in the data. The F1 score is the harmonic mean of precision and recall and provides a balanced measure of the model's performance.

7.1. Logistic regression (initial prediction)

We used Multinomial logistic regression for our initial prediction. It is a statistical model used to analyse the relationship between a categorical dependent variable and one or more independent variables.
For the logistic regression, we observed that late flights had a precision, recall, and F1 score of 0, 0, 0 indicating that the model did not accurately predict these classes. However, for early, on time and severely late flights, the model achieved a precision, recall, and F1 score of 0.38, 0.46, and 0.42, and 0.41, 0.62, and 0.49 and 0.4, 0.05, 0.08 respectively. Overall the logistic regression had an accuracy of 0.40.

```
X = data.drop('Status', axis=1)
y = data['Status']
X_train_multi, X_test_multi, y_train_multi, y_test_multi = train_test_split(X, y, test_size=0.2, random_state=42)

model = LogisticRegression(multi_class='multinomial', solver='lbfgs')
# Train the model
model.fit(X_train_multi, y_train_multi)
# Make predictions on the testing set
y_pred = model.predict(X_test_multi)
# Evaluate the performance of the model using accuracy score
accuracy = accuracy_score(y_test_multi, y_pred)
# Compute the confusion matrix
cm = confusion_matrix(y_test_multi, y_pred)
print('Accuracy:', accuracy)
print('Confusion Matrix:')
print(cm)
print(classification_report(y_test_multi, y_pred))
```

```
: LogisticRegression(multi_class='multinomial')

Accuracy: 0.40293040293040294
Confusion Matrix:
[[36  0 41  1]
 [ 5  0 29  1]
 [44  0 72  1]
 [ 9  0 32  2]]
               precision    recall  f1-score   support

        Early       0.38      0.46      0.42        78
         Late       0.00      0.00      0.00        35
      On-time       0.41      0.62      0.49       117
Severely late       0.40      0.05      0.08        43

     accuracy                           0.40       273
    macro avg       0.30      0.28      0.25       273
 weighted avg       0.35      0.40      0.34       273
```

7.2. Random Forest:

We used Random Forest, which is an ensemble learning method that builds multiple decision trees and combines their predictions to make a final prediction. Each tree is trained on a randomly selected subset of the training data, and at each split, the algorithm chooses the best feature from a random subset of features.

For the Random Forest model, we observed that severely late and late flights had a precision, recall, and F1 score of 0, 0, and 0, indicating that the model did not accurately predict these classes. However, for on-time and early flights, the model achieved a precision, recall, and F1 score of 0.47, 0.92, and 0.62, and 0.72, 0.33, and 0.46, respectively. Overall, the Random Forest model had an accuracy of 0.50.

In [174]:
```
#Random Forest Train
accuracy = metrics.accuracy_score(y_train, y_pred_train)
print("Accuracy: {:.2f}".format(accuracy))
cm=confusion_matrix(y_train,y_pred_train)
print('Confusion Matrix: \n', cm)
print(classification_report(y_train, y_pred_train, target_names=class_names))
```

```
Accuracy: 0.50
Confusion Matrix:
 [[  0 152  12   0]
 [  0 517  14   0]
 [  0 289 106   0]
 [  0 144   4   0]]
               precision    recall  f1-score   support

Severely late       0.00      0.00      0.00       164
      On-time       0.47      0.97      0.63       531
        Early       0.78      0.27      0.40       395
         Late       0.00      0.00      0.00       148

     accuracy                           0.50      1238
    macro avg       0.31      0.31      0.26      1238
 weighted avg       0.45      0.50      0.40      1238
```

In [176]:
```
#Random Forest Test
accuracy = metrics.accuracy_score(y_test, y_pred_test)
print("Accuracy: {:.2f}".format(accuracy))

from sklearn.metrics import confusion_matrix, classification_report
cm=confusion_matrix(y_test,y_pred_test)
print('Confusion Matrix: \n', cm)
print(classification_report(y_test, y_pred_test, target_names=class_names))
```

```
Accuracy: 0.50
Confusion Matrix:
 [[ 0 16  0  0]
 [ 0 49  4  0]
 [ 0 26 13  0]
 [ 0 14  1  0]]
               precision    recall  f1-score   support

Severely late       0.00      0.00      0.00        16
      On-time       0.47      0.92      0.62        53
        Early       0.72      0.33      0.46        39
         Late       0.00      0.00      0.00        15

     accuracy                           0.50       123
    macro avg       0.30      0.31      0.27       123
 weighted avg       0.43      0.50      0.41       123
```

7.3. K-Nearest Neighbors (KNN):

We also evaluated KNN, which is a non-parametric classification algorithm that assigns a class label to a data point based on the k-nearest neighbors in the training data. The value of k determines the number of neighbors that are considered, and the class label is assigned based on the majority class among the k-nearest neighbors.

For the KNN model, we observed that severely late flights had a precision, recall, and F1 score of 0.20, 0.06, and 0.10, while late flights had a precision, recall, and F1 score of 0, 0, and 0. For on-time and early flights, the model achieved a precision, recall, and F1 score of 0.44, 0.72, and 0.54, and 0.40, 0.31, and 0.35, respectively. The KNN model had an overall accuracy of 0.41.

```
In [183]: #KNN Test
          accuracy = metrics.accuracy_score(y_test, knn_pred)
          print("Accuracy: {:.2f}".format(accuracy))

          from sklearn.metrics import confusion_matrix, classification_report
          cm=confusion_matrix(y_test,knn_pred)
          print('Confusion Matrix: \n', cm)
          print(classification_report(y_test, knn_pred, target_names=class_names))

          Accuracy: 0.41
          Confusion Matrix:
           [[ 1 11  4  0]
           [ 2 38 12  1]
           [ 1 26 12  0]
           [ 1 12  2  0]]
                          precision    recall  f1-score   support

          Severely late       0.20      0.06      0.10        16
                On-time       0.44      0.72      0.54        53
                  Early       0.40      0.31      0.35        39
                   Late       0.00      0.00      0.00        15

               accuracy                           0.41       123
              macro avg       0.26      0.27      0.25       123
           weighted avg       0.34      0.41      0.36       123
```

7.4. Gradient Boosting:

Lastly, we evaluated Gradient Boosting, which is an ensemble learning method that builds multiple weak learners in a sequential manner, where each learner tries to correct the mistakes of the previous learner. The final prediction is made by combining the predictions of all the learners.

For the Gradient Boosting model, we observed that it had an overall accuracy of 0.46, which means that it correctly classified 46% of the flights in the test data set.

```
In [185]: test_output = pd.DataFrame(gb.predict(xrf_test), index = xrf_test.index, columns = ['pred_Y'])
          test_output = test_output.merge(y_test, left_index = True, right_index = True)
          test_output.head()
          print('Fraction of correct classification ')
          gb.score(xrf_test, y_test)
```

Out[185]:

	pred_Y	Status
268	1	2
920	2	2
799	2	2
512	1	1
1202	1	3

```
Fraction of correct classification
```

Out[185]: 0.4634146341463415

Based on the accuracy values we obtained, Random Forest had the highest accuracy of 0.50, followed by Gradient Boosting with an accuracy of 0.46 and KNN with an accuracy of 0.41.

8. Predictions

8.1. Initial prediction:

The below image shows a CSV file containing flight data where the status of the flights was predicted for the dates April 12 to April 15, with the last column indicating the actual status.

Date	Day	Origin Airport	Flight Numb	Arrival Time	Predicted Status	Actual Status (Early, On-time, Late, Severly Late)
4/12/23	Wednesday	ORD	UA 3839	10:00 AM	Early	Early
4/12/23	Wednesday	ORD	UA 3524	4:52 PM	Early	On-Time
4/12/23	Wednesday	ORD	UA 538	9:34 PM	Early	On-Time
4/13/23	Thursday	ORD	UA 3839	10:00 AM	Early	Late
4/13/23	Thursday	ORD	UA 3524	4:50 PM	Early	Early
4/13/23	Thursday	ORD	UA 538	9:34 PM	Early	On-Time
4/14/23	Friday	ORD	UA 3839	10:00 AM	Early	On-Time
4/14/23	Friday	ORD	UA 3524	4:50 PM	Early	Early
4/14/23	Friday	ORD	UA 538	9:34 PM	Early	Early
4/15/23	Saturday	ORD	UA 3839	10:00 AM	Early	Early
4/15/23	Saturday	ORD	UA 3524	4:50 PM	Early	Early
4/15/23	Saturday	ORD	UA 538	9:34 PM	Early	Severely Late
4/12/23	Wednesday	DEN	UA 604	3:12 PM	On-time	Severely Late
4/13/23	Thursday	DEN	UA 604	3:12 PM	On-time	On-Time
4/14/23	Friday	DEN	UA 604	3:12 PM	On-time	On-Time
4/15/23	Saturday	DEN	UA 604	3:12 PM	On-time	Severely Late
4/12/23	Wednesday	EWR	UA 4189	10:46 AM	On-time	Early
4/12/23	Wednesday	EWR	UA 1412	11:42 PM	On-time	Early
4/13/23	Thursday	EWR	UA 4189	10:46 AM	On-time	Early
4/13/23	Thursday	EWR	UA 1412	11:42 PM	On-time	Early
4/14/23	Friday	EWR	UA 4189	10:46 AM	On-time	Early
4/14/23	Friday	EWR	UA 1412	11:42 PM	On-time	On-Time
4/15/23	Saturday	EWR	UA 4189	10:46 AM	On-time	Early
4/15/23	Saturday	EWR	UA 1412	11:17 PM	On-time	Severely Late
4/12/23	Wednesday	IAD	UA 4490	1:57 PM	On-time	Early
4/12/23	Wednesday	IAD	UA 4165	6:59 PM	On-time	On-time
4/13/23	Thursday	IAD	UA 4490	1:57 PM	On-time	On-Time
4/13/23	Thursday	IAD	UA 4165	6:59 PM	On-time	Early
4/14/23	Friday	IAD	UA 4490	1:57 PM	On-time	On-Time
4/14/23	Friday	IAD	UA 4165	6:59 PM	On-time	Early
4/15/23	Saturday	IAD	UA 3805	1:58 PM	On-time	Late
4/15/23	Saturday	IAD	UA 4165	6:59 PM	On-time	Canceled

8.2 Second prediction:

The below image shows a CSV file containing flight data where the status of the flights was predicted for the dates April 21 to April 24, with the last column indicating the actual status.

Date	Day	Origin Airpor	Flight Numb	Arrival Time	Predicted Status (Early, C	Actual Status (Early, On-time, Late, Sev
4/21/23	Friday	ORD	UA 3839	10:00 AM	On-time	Early
4/21/23	Friday	ORD	UA 3524	4:50 PM	Early	Severely Late
4/21/23	Friday	ORD	UA 538	9:34 PM	On-time	Early
4/22/23	Saturday	ORD	UA 3839	10:00 AM	On-time	On-time
4/22/23	Saturday	ORD	UA 3524	4:50 PM	On-time	On-time
4/22/23	Saturday	ORD	UA 538	9:34 PM	On-time	On-time
4/23/23	Sunday	ORD	UA 3839	10:00 AM	On-time	Severely Late
4/23/23	Sunday	ORD	UA 3524	4:55 PM	On-time	Early
4/23/23	Sunday	ORD	UA 538	9:34 PM	On-time	Severely Late
4/24/23	Monday	ORD	UA 3839	10:00 AM	On-time	On-time
4/24/23	Monday	ORD	UA 3524	4:50 PM	On-time	Early
4/24/23	Monday	ORD	UA 538	9:34 PM	On-time	Early
4/21/23	Friday	DEN	UA 604	3:12 PM	On-time	Early
4/22/23	Saturday	DEN	UA 604	3:12 PM	On-time	Late
4/23/23	Sunday	DEN	UA 604	3:12 PM	On-time	On-time
4/24/23	Monday	DEN	UA 604	3:12 PM	On-time	On-time
4/21/23	Friday	EWR	UA 4189	10:46 AM	On-time	Early
4/21/23	Friday	EWR	UA 1412	11:42 PM	On-time	Late
4/22/23	Saturday	EWR	UA 4189	10:46 AM	On-time	Severely Late
4/22/23	Saturday	EWR	UA 1412	11:17 PM	On-time	Severely Late
4/23/23	Sunday	EWR	UA 4189	10:46 AM	On-time	Early
4/23/23	Sunday	EWR	UA 1412	11:42 PM	On-time	On-time
4/24/23	Monday	EWR	UA 4189	10:46 AM	On-time	Early
4/24/23	Monday	EWR	UA 1412	11:42 PM	On-time	Early
4/21/23	Friday	IAD	UA 4490	1:57 PM	On-time	On-time
4/21/23	Friday	IAD	UA 4165	6:59 PM	On-time	On-time
4/22/23	Saturday	IAD	UA 3805	1:58 PM	On-time	On-time
4/22/23	Saturday	IAD	UA 4165	6:59 PM	On-time	Severely Late
4/23/23	Sunday	IAD	UA 4490	1:57 PM	On-time	Early
4/23/23	Sunday	IAD	UA 4165	6:59 PM	On-time	On-time
4/24/23	Monday	IAD	UA 4490	1:57 PM	On-time	On-time
4/24/23	Monday	IAD	UA 4165	6:59 PM	On-time	Early

9. Conclusion:

In conclusion, we evaluated the performance of four machine learning models for predicting aircraft delays using various evaluation metrics. The Random Forest model achieved the highest accuracy of 0.50, while the KNN model had an accuracy of 0.41 and Gradient Boosting had 0.46.

However, there are several techniques that we can implement in future to potentially increase the models accuracy. These may include:

1. Feature Engineering: We could explore different features or combinations of features to include in the model. This could involve creating new features from the existing ones, selecting only the most important features using feature selection techniques, or using domain knowledge to engineer new features that could better capture the patterns in the data.

2. Hyperparameter Tuning: Each machine learning algorithm has a set of hyperparameters that can be tuned to optimize the model's performance. We could use techniques such as grid search, randomized search, or Bayesian optimization to find the best combination of hyperparameters for each model.

3. Additional Data for Training - Testing: By adding more data, we could potentially increase the amount of information available to the models, which could lead to better performance. Additionally, adding more data could help to balance the representation of the different flight classes, which could improve the model's ability to predict each class accurately.

4. Data Balancing: As we noticed, the severely late and late flights were underrepresented in our dataset, which could negatively impact the model's performance. We could use techniques such as oversampling, undersampling, or adjusting the class weights to balance the data and improve the model's performance.

5. Model Selection: We could also explore other machine learning algorithms that may perform better on this particular dataset. There are many other algorithms available, including support vector machines, neural networks among others.

By exploring these techniques, we could potentially improve the accuracy of all three models and achieve better performance in predicting aircraft delays.

10. References:

https://www.transtats.bts.gov/ontime/Arrivals.aspx
https://www.weatherbit.io/api
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
https://ieeexplore.ieee.org/document/8389254
https://ieeexplore.ieee.org/document/8272656
https://ieeexplore.ieee.org/document/8876970