Syracuse University

PROJECT REPORT

Introduction to Machine Learning

TEAM MEMBERS:

Bhumika Murali

Varshini Narayana

Manjunath Sangappa

Akhilesh Jadhav

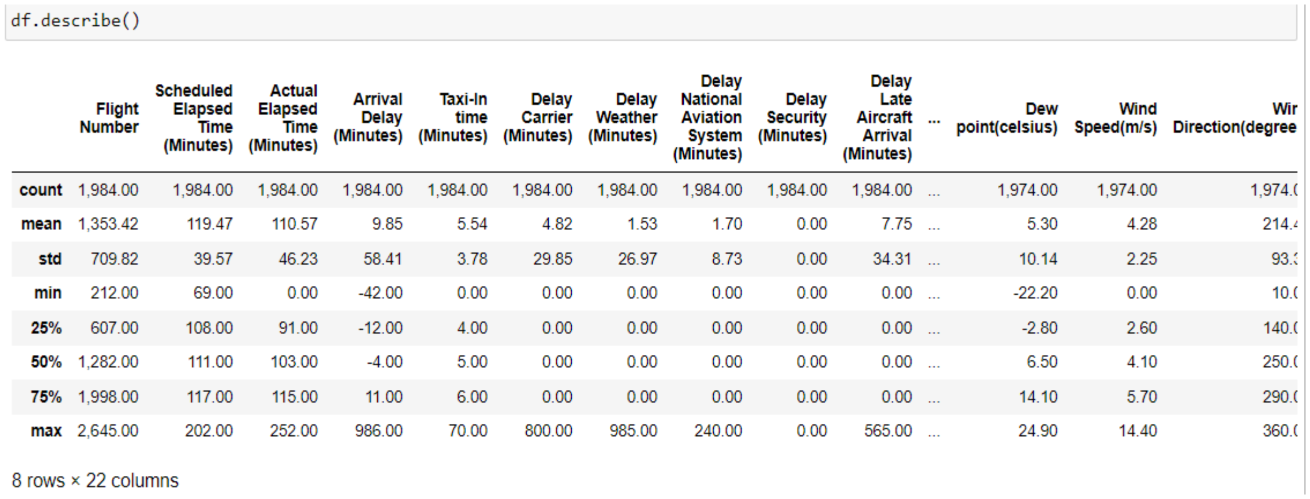
Introduction:

The United Airlines direct flights into Syracuse (SYR) from four cities are the subject of this project. The aim is to predict if each flight's arrival time into SYR will be early, on-time, delayed, or severely delayed, 1-4 days in advance. There are no restrictions on the data sources, and weather predictions for the next few days can be considered, as weather is one factor that can cause airline delays. The final predictions will be made on a CSV file with a column for predicting the arrival time, date (April 21-24 to be predicted on April 20), origin airport, and flight number. Ground truth data will be based on the United Airlines website, where flights that are more than 10 minutes early will be considered early, within 10 minutes (plus or minus) of the scheduled time will be considered on-time, more than 10 minutes late but up to 30 minutes late will be considered late, and anything beyond 30 minutes late will be considered severely late. Any approach using regression or classification, including statistical analysis and time series, would be reasonable choices for the methodology, but neural networks or deep learning-based methods are not allowed.

Exploratory Data Analysis:

As part of our data analysis project, we collected a dataset on flight arrivals from "https://www.transtats.bts.gov/ontime/" that includes various features such as the date, origin airport, scheduled arrival time, actual arrival time, and arrival delay. Before we can start building models or doing any meaningful analysis, we need to clean and preprocess the data to ensure that it's in a usable format.

First, we loaded the data from an Excel file and checked for any missing values using the isna() function. We found that there were some missing values in the dataset, so we dropped those rows using the dropna() function to ensure that our dataset only contains complete records. We have used df.describe() which will provides summary statistics of the numerical columns in the DataFrame df, such as count, mean, standard deviation, minimum and maximum values, and quartile values.

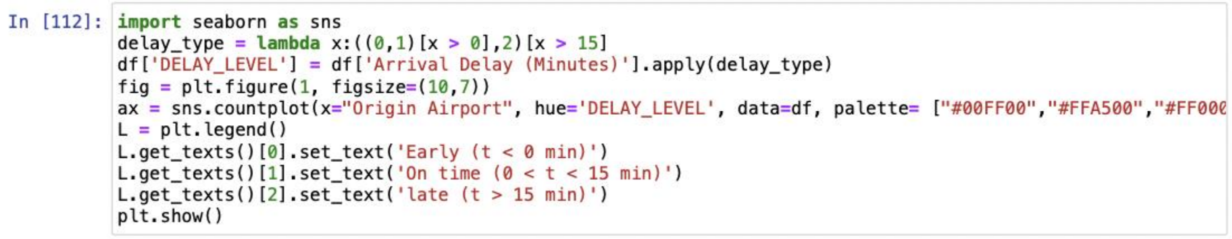


```
df.describe()
```

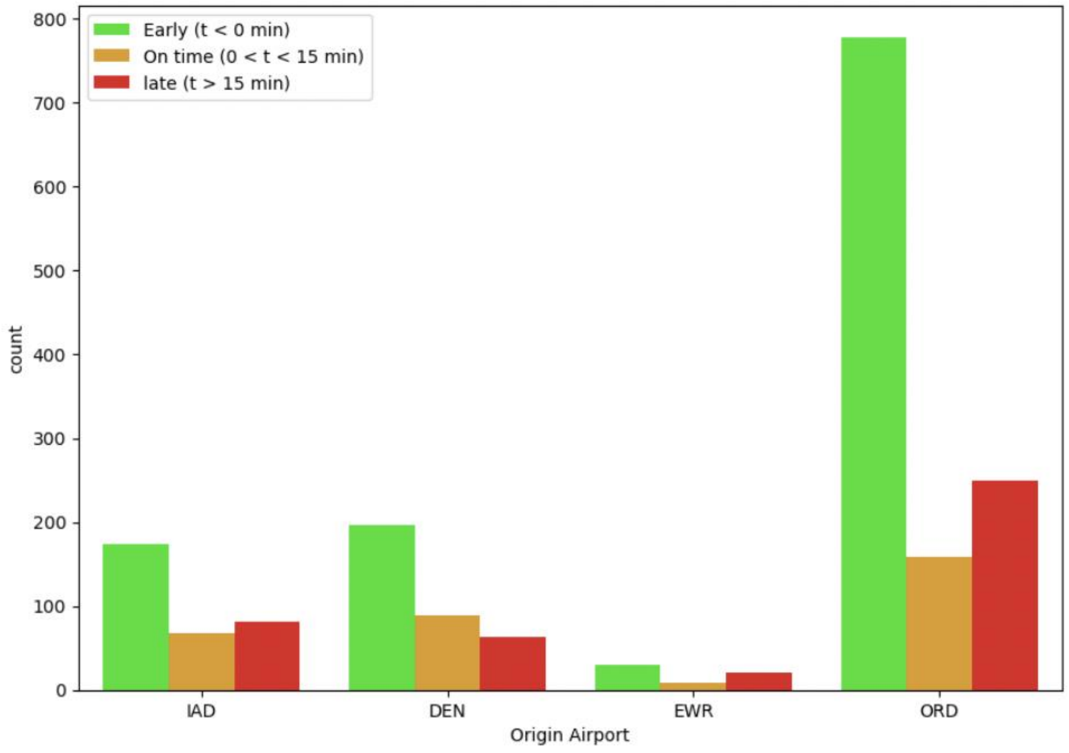
	Flight Number	Scheduled Elapsed Time (Minutes)	Actual Elapsed Time (Minutes)	Arrival Delay (Minutes)	Taxi-In time (Minutes)	Delay Carrier (Minutes)	Delay Weather (Minutes)	Delay National Aviation System (Minutes)	Delay Security (Minutes)	Delay Late Aircraft Arrival (Minutes)	...	Dew point(celsius)	Wind Speed(m/s)	Wir Direction(degree
count	1,984.00	1,984.00	1,984.00	1,984.00	1,984.00	1,984.00	1,984.00	1,984.00	1,984.00	1,984.00	...	1,974.00	1,974.00	1,974.
mean	1,353.42	119.47	110.57	9.85	5.54	4.82	1.53	1.70	0.00	7.75	...	5.30	4.28	214.
std	709.82	39.57	46.23	58.41	3.78	29.85	26.97	8.73	0.00	34.31	...	10.14	2.25	93.
min	212.00	69.00	0.00	-42.00	0.00	0.00	0.00	0.00	0.00	0.00	...	-22.20	0.00	10.
25%	607.00	108.00	91.00	-12.00	4.00	0.00	0.00	0.00	0.00	0.00	...	-2.80	2.60	140.
50%	1,282.00	111.00	103.00	-4.00	5.00	0.00	0.00	0.00	0.00	0.00	...	6.50	4.10	250.
75%	1,998.00	117.00	115.00	11.00	6.00	0.00	0.00	0.00	0.00	0.00	...	14.10	5.70	290.
max	2,645.00	202.00	252.00	986.00	70.00	800.00	985.00	240.00	0.00	565.00	...	24.90	14.40	360.

8 rows × 22 columns

Next, we noticed that some of the columns in the dataset were not useful for our analysis, such as the carrier code, flight number, and tail number. We dropped these columns using the drop() function to create a subset of the original dataset that only contains relevant features.We also found the correlation between the features and the delay level with respect to the origin airport DEN, EWR, IAD, ORD). We will use these findings in the next part to discuss feature engineering.



```
In [112]: import seaborn as sns
          delay_type = lambda x:((0,1)[x > 0],2)[x > 15]
          df['DELAY_LEVEL'] = df['Arrival Delay (Minutes)'].apply(delay_type)
          fig = plt.figure(1, figsize=(10,7))
          ax = sns.countplot(x="Origin Airport", hue='DELAY_LEVEL', data=df, palette= ["#00FF00","#FFA500","#FF000
          L = plt.legend()
          L.get_texts()[0].set_text('Early (t < 0 min)')
          L.get_texts()[1].set_text('On time (0 < t < 15 min)')
          L.get_texts()[2].set_text('late (t > 15 min)')
          plt.show()
```



Preprocessing:

As a team, we started by examining the data types of each column in the dataset using the dtypes method. We noticed that some columns contained irrelevant or redundant information, so we dropped them from our analysis. Specifically, we removed the columns Carrier Code, Flight Number, Tail Number, Wheels-on Time, and Weather Description.

Next, we converted the Date (MM/DD/YYYY) column to a datetime type and created a new column to extract the day of the week. We also extracted the hour and minute components from the Scheduled Arrival Time, Actual Arrival Time, Actual departure time, and Scheduled departure time columns to better understand the time-related features of the dataset.

After creating these new columns, we dropped the original time-related columns that we had extracted the hour and minute components from, as well as a dummy column that we didn't need. We also used the get_dummies function from the Pandas library to convert categorical variables into numerical form, as machine learning algorithms typically require numerical data as input.

Finally, we added a new column Day_Friday and set it to zero before converting it to an unsigned integer data type. We also split the data into our features X_train and our target variable y_train, with X_train including all columns except Scheduled_Arrival_hours and Scheduled_Arrival_minutes. We plan to use these features to train our machine learning model to predict the Scheduled_Arrival_minutes target variable.

MODELS USED:

Linear Regressor:

Linear Regression is a type of supervised learning algorithm in which the goal is to predict a continuous output variable (also called a dependent variable) based on one or more input variables (also called independent variables). The basic intuition behind Linear Regression is to find a linear relationship between the input variables and the output variable.

The formula for Linear Regression can be expressed as:

$y=mx+b$

Here in our project, linear regression is used in the for flight delay prediction to create a model that predicts the delay time of a flight based on various factors such as weather conditions, airline, time of day, and previous delay history. The model was applied on our dataset containing information about various flights, including their scheduled departure times, actual departure times, airline, origin, destination, weather conditions, and other relevant factors. We used this data to train a linear regression model that can predict the delay time of a flight based on these factors.

Once the model is trained, we use it to predict the delay time of new flights based on their characteristics. We evaluated the accuracy of the model using the testing set and made necessary adjustments to improve its performance.

The results predicted by the linear regression model were not that accurate.

Lasso and Ridge Regressor:

Both are similar linear regression techniques used for predicting a continuous output variable. They are used to address the problem of overfitting and improve the generalisation of the model

Lasso regression, also known as L1 regularisation, works by adding a penalty term proportional to the absolute value of the regression coefficients. This helps to reduce the impact of less important features on the model and encourages sparse solutions where only a subset of features have non-zero coefficients. In the case of flight delay prediction, Lasso regression can be used to identify the most important factors that contribute to the delay time of a flight.

Ridge regression, also known as L2 regularisation, works by adding a penalty term proportional to the square of the regression coefficients. This helps to reduce the magnitude of the regression coefficients and prevent overfitting by encouraging solutions with smaller coefficients. In the case of flight delay prediction, Ridge regression can be used to smooth out the impact of highly correlated features and improve the stability of the model.

Lasso regression is generally preferred when there are many irrelevant features in the data, while Ridge regression is preferred when there are highly correlated features that need to be taken into account.

Overall, the results obtained by lasso and ridge regression models were good.

Decision Tree Regressor:

Decision Tree Regression is a type of supervised learning algorithm that is used for predicting a continuous output variable based on a set of input variables. The basic intuition behind Decision Tree Regressor is to recursively split the input space into regions, based on the values of the input variables, in a way that minimises the variance of the output variable in each region.

In this algorithm, the input space is divided into a series of binary decisions that eventually lead to a prediction for the output variable. The decisions are made by choosing the input variable that best splits the data into subsets with the smallest possible variance. This process is repeated recursively for each subset until a stopping criterion is reached.

The advantage of using Decision Tree Regressor is that it is simple to understand and interpret, and can handle non-linear relationships between the input variables and the output variable. It is also robust to noisy data and missing values.

However, Decision Tree Regressor can be prone to overfitting, particularly when the tree is too deep or when the input space is too complex. This can lead to poor generalisation of the model to new data.
Overall, Decision Tree Regressor is a simple and effective algorithm that can be used for regression tasks with moderate complexity. However, it did not perform as well as Random Forest Regressor on more complex or high-dimensional data.

Random Forest Regressor:

Random Forest Regressor is also a type of supervised learning algorithm that is used for predicting a continuous output variable. It is a powerful and popular ensemble method that combines multiple decision trees to make accurate predictions. The basic intuition behind Random Forest Regressor is to create a large number of decision trees, each trained on a random subset of the data, and then combine their predictions to obtain a more accurate and robust prediction.

This algorithm is able to handle non-linear relationships between the input variables and the output variable, which makes it more powerful than linear regression. The ensemble method also helps to reduce overfitting and improve the generalisation of the model. Overall, Random Forest Regressor is a powerful and flexible algorithm that can be used for a wide range of regression tasks.

The advantage of using Random Forest is that it can handle high-dimensional data with a large number of features, as well as noisy and missing data. It also helps to prevent overfitting and improve the accuracy of predictions compared to a single decision tree.

Overall, Random Forest is a powerful and widely used algorithm that can be applied to a variety of machine-learning problems. The results obtained from this model were also more accurate compared to other models.

Results and Discussions:

Our project aimed to make predictions about flight delays using data from the Bureau of Transportation website.

We divided the Bureau of Transportation data into train and test sets. We then ran the data through various models, including linear regression, decision tree regression, random forest regression, and other models as discussed previously.

After comparing the results, we found that the Random Forest Regressor was the most effective model for predicting flight delays. The mean error was lower when compared to other models, which indicated that the Random Forest Regressor was better at making accurate predictions.

For Initial Prediction, we analyzed flight data from a 6-7 month period, which included details such as the date, day, origin airport, and flight number whereas we excluded variables such as 'Scheduled Elapsed Time (Minutes)', 'Actual Elapsed Time (Minutes)', 'Arrival Delay (Minutes)', 'Taxi-In time (Minutes)', 'Delay Carrier (Minutes)', 'Delay Weather (Minutes)', 'Delay National Aviation System (Minutes)', 'Delay Security (Minutes)', and 'Delay Late Aircraft Arrival (Minutes)'. This was because we did not have this information for upcoming flights in our test data. However, we found that these variables were not enough to make accurate predictions.

After getting to know the actual initial predictions, we used the actual arrival status of the flights to check our model's accuracy. We compared our predictions against the actual arrival status to determine how similar they were.

We found that the Random Forest Regressor showed the most similarity between our predictions and the actual arrival status. This further supported our previous findings that the Random Forest Regressor was the most effective model for predicting flight delays.

We decided to collect additional weather data using an API to improve our model. To incorporate the weather data into our analysis, we merged it with our train and test data in Excel. We used the VLOOKUP feature of Excel to match the weather data with the corresponding flights in our dataset.

By merging the weather data into our dataset, we were able to include more relevant variables in our analysis. This allowed us to gather information such as pressure, temperature, dew point, wind speed and direction, precipitation, snowfall, cloud cover, humidity, solar radiation, UV index, visibility, weather description, and day/night indicators. We then increased the size of our dataset by collecting 6 years of data from the Bureau of Transportation website. After preprocessing the data and running it through several models, we found that the Random Forest Regressor provided the best results for predicting flight delays.

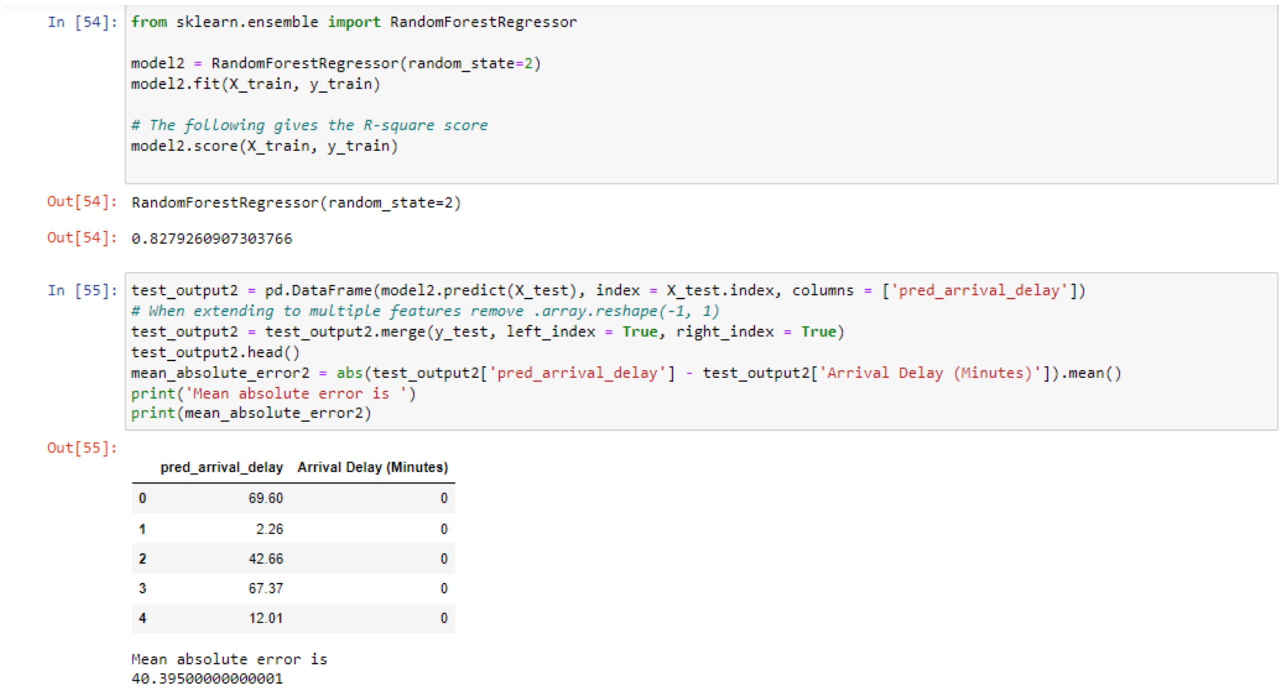
Below are the model's results for Random Forest Regressor:

R-Sq Score = 0.83

MAE = 40.39

The performance of the Random Forest Regressor was compared to other models during both the initial prediction and testing stages. In both cases, the Random Forest Regressor outperformed the other models, indicating that it was the most effective model for predicting flight delays.

Below are the results:

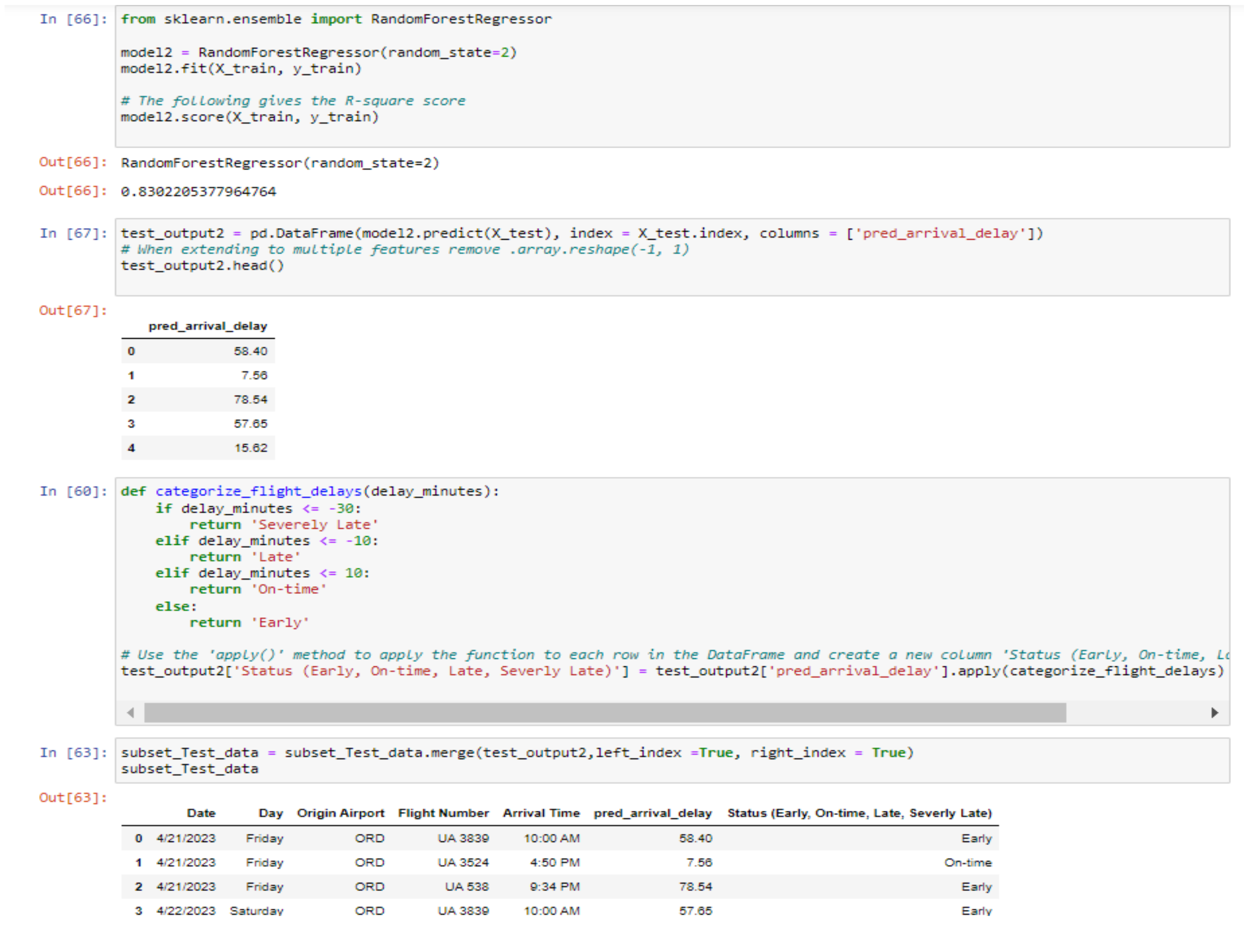


```
In [54]: from sklearn.ensemble import RandomForestRegressor

         model2 = RandomForestRegressor(random_state=2)
         model2.fit(X_train, y_train)

         # The following gives the R-square score
         model2.score(X_train, y_train)

Out[54]: RandomForestRegressor(random_state=2)

Out[54]: 0.8279260907303766

In [55]: test_output2 = pd.DataFrame(model2.predict(X_test), index = X_test.index, columns = ['pred_arrival_delay'])
         # When extending to multiple features remove .array.reshape(-1, 1)
         test_output2 = test_output2.merge(y_test, left_index = True, right_index = True)
         test_output2.head()
         mean_absolute_error2 = abs(test_output2['pred_arrival_delay'] - test_output2['Arrival Delay (Minutes)']).mean()
         print('Mean absolute error is ')
         print(mean_absolute_error2)

Out[55]:
```

	pred_arrival_delay	Arrival Delay (Minutes)
0	69.60	0
1	2.26	0
2	42.66	0
3	67.37	0
4	12.01	0

```
Mean absolute error is
40.39500000000001
```

The following code snippet shows how we made final predictions using the Random Forest Regressor.



```
In [66]: from sklearn.ensemble import RandomForestRegressor

         model2 = RandomForestRegressor(random_state=2)
         model2.fit(X_train, y_train)

         # The following gives the R-square score
         model2.score(X_train, y_train)

Out[66]: RandomForestRegressor(random_state=2)

Out[66]: 0.8302205377964764

In [67]: test_output2 = pd.DataFrame(model2.predict(X_test), index = X_test.index, columns = ['pred_arrival_delay'])
         # When extending to multiple features remove .array.reshape(-1, 1)
         test_output2.head()

Out[67]:
```

	pred_arrival_delay
0	58.40
1	7.56
2	78.54
3	57.65
4	15.62

```
In [60]: def categorize_flight_delays(delay_minutes):
             if delay_minutes <= -30:
                 return 'Severely Late'
             elif delay_minutes <= -10:
                 return 'Late'
             elif delay_minutes <= 10:
                 return 'On-time'
             else:
                 return 'Early'

         # Use the 'apply()' method to apply the function to each row in the DataFrame and create a new column 'Status (Early, On-time, La
         test_output2['Status (Early, On-time, Late, Severly Late)'] = test_output2['pred_arrival_delay'].apply(categorize_flight_delays)

In [63]: subset_Test_data = subset_Test_data.merge(test_output2,left_index =True, right_index = True)
         subset_Test_data

Out[63]:
```

	Date	Day	Origin Airport	Flight Number	Arrival Time	pred_arrival_delay	Status (Early, On-time, Late, Severly Late)
0	4/21/2023	Friday	ORD	UA 3839	10:00 AM	58.40	Early
1	4/21/2023	Friday	ORD	UA 3524	4:50 PM	7.56	On-time
2	4/21/2023	Friday	ORD	UA 538	9:34 PM	78.54	Early
3	4/22/2023	Saturday	ORD	UA 3839	10:00 AM	57.65	Early

Overall, our approach involved expanding the scope of our data to include more relevant variables, increasing the size of our dataset, and testing various models to find the one that provided the most accurate predictions. By doing so, we improved our model's effectiveness and provided more accurate predictions about flight delays.