



FLIGHT DELAYS PREDICTION USING MACHINE LEARNING

Flight Delay Prediction Using Machine Learning

Our project aims to build a Machine Learning model integrated with a Flask application to predict flight delays. By analyzing historical flight data, weather conditions, airport congestion, and other factors, we develop a predictive model. This model is then incorporated into a user-friendly Flask web application, allowing travelers and airlines to anticipate potential delays and plan accordingly, enhancing travel experiences and operational efficiency.

Technical Architecture:

Project Flow:

- To User interacts with the UI (User Interface) to enter Data
- The entered data is analyzed by the model which is integrated
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection.
 - Collect the dataset or create the dataset
- Data Preprocessing.
 - Import the Libraries.
 - Importing the dataset.
 - Checking for Null Values.
 - Data Visualization.
 - Taking care of Missing Data.
 - Label encoding.
 - One Hot Encoding.
 - Feature Scaling.
 - Splitting Data into Train and Test.
- Model Building
 - Training and testing the model
 - Evaluation of Model
- Application Building
 - Create an HTML file
 - Build a Python Code

Project Demonstration & Documentation

- a. Record explanation Video for project end to end solution
 - Project Documentation-Step by step project development procedure

Prior Knowledge:

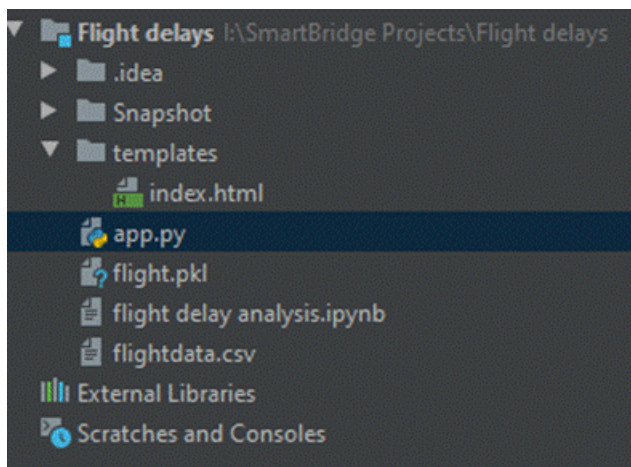
You must have prior knowledge of following topics to complete this project.

- ML Concepts
 - o Supervised learning: <https://www.javatpoint.com/supervised-machine-learning>
 - o Unsupervised learning: <https://www.javatpoint.com/unsupervised-machine-learning>
- Decision tree: <https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm>
- Random forest: <https://www.javatpoint.com/machine-learning-random-forest-algorithm>

- KNN: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
- XgBoost: <https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/>
- Evaluation metrics: <https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/>
- Flask Basics: https://www.youtube.com/watch?v=Ij4I_CvBnt0

Project Structure:

Create the Project folder which contains files as shown below



- The project folder contains the following folders
 - flightdata.csv is the dataset used for training our model.
 - flight delay analysis.ipynb is the python file where the ML algorithm is applied to the dataset for testing and training. Finally, the model is saved for future use.
 - flight.pkl is the saved model used in the flask to predict the output instantly for the given inputs.
 - To build a Flask Application, save HTML pages in the templates folder and a python script app.py for server-side scripting.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

Airlines face significant operational and financial challenges due to unexpected flight delays, leading to customer

dissatisfaction, increased costs, and scheduling disruptions. There is a need for a predictive system that can accurately forecast flight delays in advance using historical and real-time flight data, enabling airlines to make informed decisions, improve efficiency, and enhance the passenger experience.

Activity 2: Business requirements

- The system should accurately predict whether a flight will be delayed or on time.
- Users must be able to input flight details such as flight number, date, origin, destination, scheduled arrival time, and departure status.
- Predictions must be provided instantly after the user submits the form.
- The result should appear in a pop-up notification with a close button for dismissal.
- The system must integrate a pre-trained machine learning model to generate predictions.
- The user interface should be visually appealing, clean, and easy to navigate.
- The system must validate user input to ensure data accuracy and completeness.
- The application should be easy to maintain and update as needed.
- The design should support scalability for future features or larger datasets.
- The system must be deployable and accessible through a standard web browser.

Activity 3: Literature Survey (Student Will Write)

Flight delays pose significant challenges to airlines and passengers, leading to economic losses and reduced customer satisfaction. Traditional statistical methods for delay prediction often fall short in capturing complex patterns. Recent studies show that machine learning algorithms like Decision Trees, Random Forests, and Neural Networks offer higher accuracy in predicting delays using features such as flight time, origin, destination, and past delay records. Public datasets like those from the U.S. Bureau of Transportation Statistics have been widely used for model training. Additionally, integrating these models into real-time, user-friendly web applications enhances their practical value for both operators and travelers.

Activity 4: Social or Business Impact.

Social Impact:

The flight delay prediction system enhances the overall passenger experience by providing timely and reliable information about flight delays. This reduces passenger anxiety, uncertainty, and inconvenience, especially for those with connecting flights or tight schedules. Improved communication fosters trust in airline services and contributes to more informed travel planning.

Business Impact:

For airlines and airport operators, the system supports better decision-making by predicting delays in advance. This leads to optimized scheduling, efficient crew and gate management, and reduced operational costs. By minimizing last-minute disruptions and enhancing punctuality, the system contributes to improved brand reputation, customer satisfaction, and profitability.

Milestone 2: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you

to download the required dataset.

Activity 1: Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: <https://www.kaggle.com/datasets/divyansh22/flight-delay-prediction>

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Importing the libraries

Import the necessary libraries as shown in the image. (optional) Here we have used visualization style.

```
import sys
import pandas as pd
import numpy as np
import seaborn as sns
import pickle
%matplotlib inline
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
import sklearn.metrics as metrics
from matplotlib import pyplot as plt
from sklearn.metrics import accuracy_score
```

Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called `read_csv()` to read the dataset. As a parameter we have to give the directory of the csv file.

- For checking the null values, `df.isna().any()` function is used. To sum those null values, we use `sum()` function. From the below image we found that there are no null values present in our dataset. So we

can skip handling the missing values step.

```
: dataset = pd.read_csv("flightdata.csv")
: dataset.head()
:
```

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	TAIL_NUM	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	...	CRS_ARR_TIME	A
0	2016	1	1	1	5	DL	N836DN	1399	10397	ATL	...	2143	
1	2016	1	1	1	5	DL	N964DN	1476	11433	DTW	...	1435	
2	2016	1	1	1	5	DL	N813DN	1597	10397	ATL	...	1215	
3	2016	1	1	1	5	DL	N587NW	1768	14747	SEA	...	1335	
4	2016	1	1	1	5	DL	N836DN	1823	14747	SEA	...	607	

5 rows × 26 columns

Activity 2: Data Preparation

As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

1. Handling missing values
2. Handling Outliers

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 2.1: Handling missing values

- For checking the null values, `dataset.isnull().sum()` function is used. To sum those null values we use `.sum()` function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

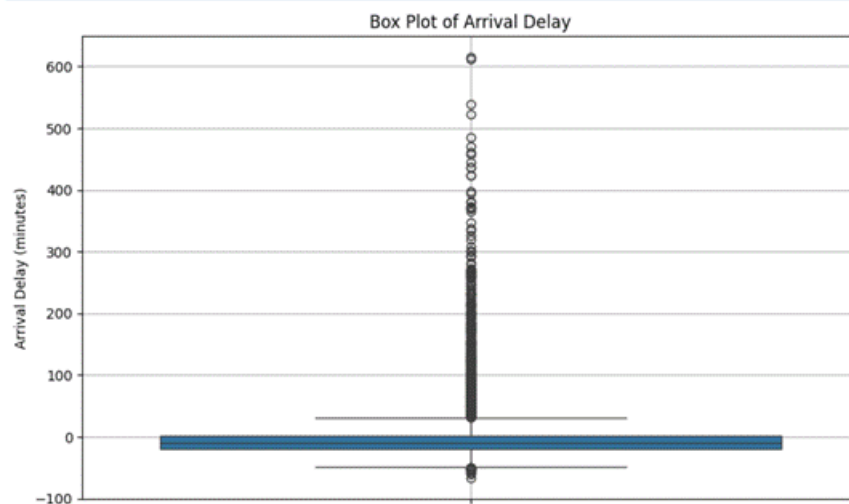
```
[8]: dataset.isnull().sum()

[8]: YEAR                0
     QUARTER             0
     MONTH              0
     DAY_OF_MONTH        0
     DAY_OF_WEEK         0
     UNIQUE_CARRIER     0
     TAIL_NUM            0
     FL_NUM              0
     ORIGIN_AIRPORT_ID   0
     ORIGIN              0
     DEST_AIRPORT_ID     0
     DEST               0
     CRS_DEP_TIME        0
     DEP_TIME           107
     DEP_DELAY           107
     DEP_DEL15           107
     CRS_ARR_TIME        0
     ARR_TIME           115
     ARR_DELAY           188
     ARR_DEL15           188
     CANCELLED           0
     DIVERTED            0
     CRS_ELAPSED_TIME    0
     ACTUAL_ELAPSED_TIME 188
     DISTANCE            0
     Unnamed: 25         11231
     dtype: int64
```

Activity 2.2: Handling Outliers

With the help of box plot, outliers are visualized. And here we are going to find upper bound and lower bound of ARR_DELAY feature with some mathematical formula.

- From the below diagram, we could visualize that ARR_DELAY feature has outliers. Box plot from sea born library is used here.



- To find upper bound we have to multiply IQR (Interquartile range) with 1.5 and add it with 3rd quartile. To find lower bound instead of adding, subtract it with 1st quartile. Take image attached below as your reference.


```

: # Step 1: Calculate IQR
Q1 = df['ARR_DELAY'].quantile(0.25)
Q3 = df['ARR_DELAY'].quantile(0.75)
IQR = Q3 - Q1

# Step 2: Define bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Step 3: Filter out outliers
df_clean = df[(df['ARR_DELAY'] >= lower_bound) & (df['ARR_DELAY'] <= upper_bound)]

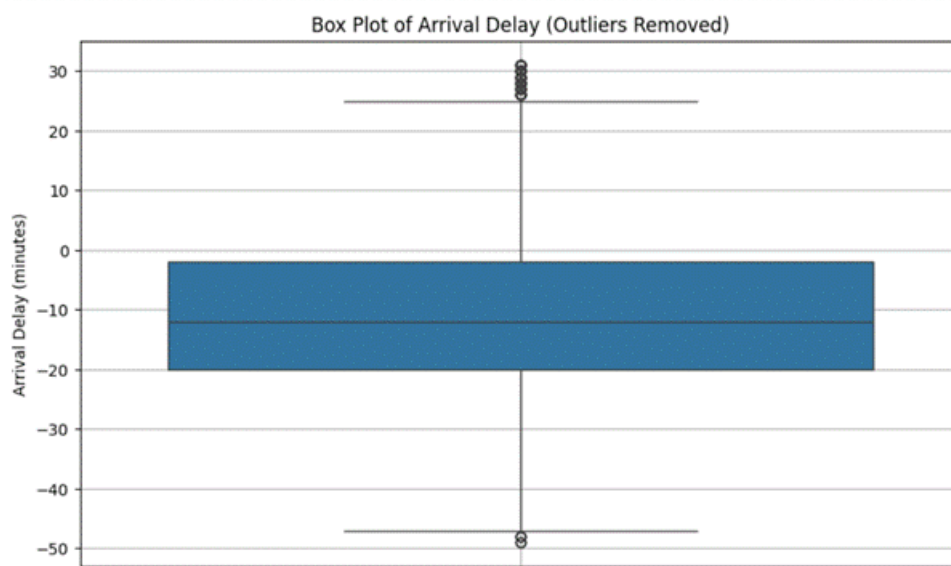
```

To handle the outliers transformation technique is used. Here log transformation is used. We have created a function to visualize the distribution and probability plot of ARR_DELAY feature.

```

# Step 4: Re-plot the cleaned boxplot
plt.figure(figsize=(10, 6))
sns.boxplot(y=df_clean['ARR_DELAY'])
plt.title("Box Plot of Arrival Delay (Outliers Removed)")
plt.ylabel("Arrival Delay (minutes)")
plt.grid(True)
plt.show()

```



Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

[7]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_NUM	ORIGIN_AIRPORT_ID	DEST_AIRPORT_ID	CRS_DEP_TIME	DEP_
count	11231.0	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11124.00
mean	2016.0	2.544475	6.628973	15.790758	3.960199	1334.325617	12334.516695	12302.274508	1320.798326	1327.18
std	0.0	1.090701	3.354678	8.782056	1.995257	811.875227	1595.026510	1601.988550	490.737845	500.30
min	2016.0	1.000000	1.000000	1.000000	1.000000	7.000000	10397.000000	10397.000000	10.000000	1.00
25%	2016.0	2.000000	4.000000	8.000000	2.000000	624.000000	10397.000000	10397.000000	905.000000	905.00
50%	2016.0	3.000000	7.000000	16.000000	4.000000	1267.000000	12478.000000	12478.000000	1320.000000	1324.00
75%	2016.0	3.000000	9.000000	23.000000	6.000000	2032.000000	13487.000000	13487.000000	1735.000000	1739.00
max	2016.0	4.000000	12.000000	31.000000	7.000000	2853.000000	14747.000000	14747.000000	2359.000000	2400.00

8 rows x 22 columns

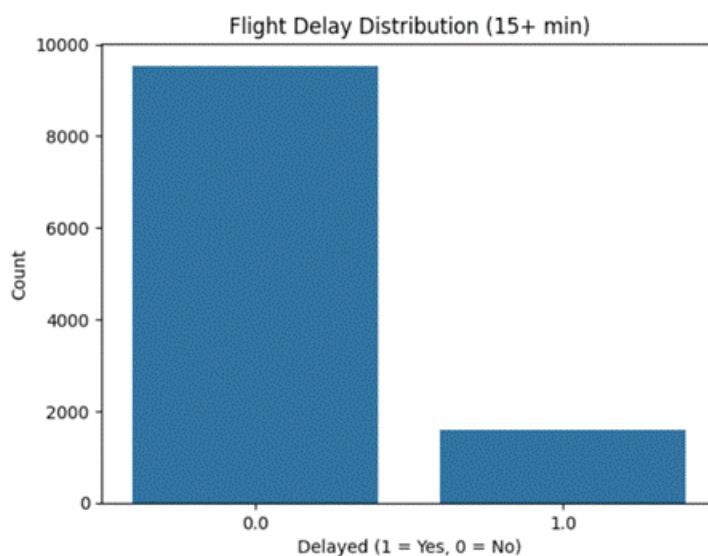
Activity 2: Visual analysis

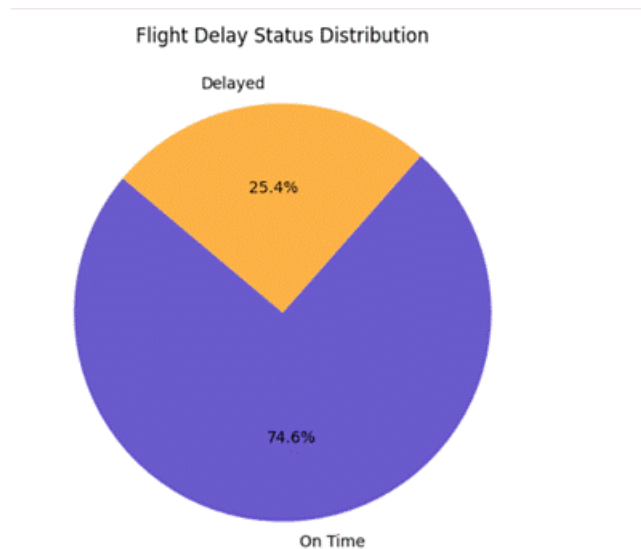
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Activity 2.1: Uni variate analysis

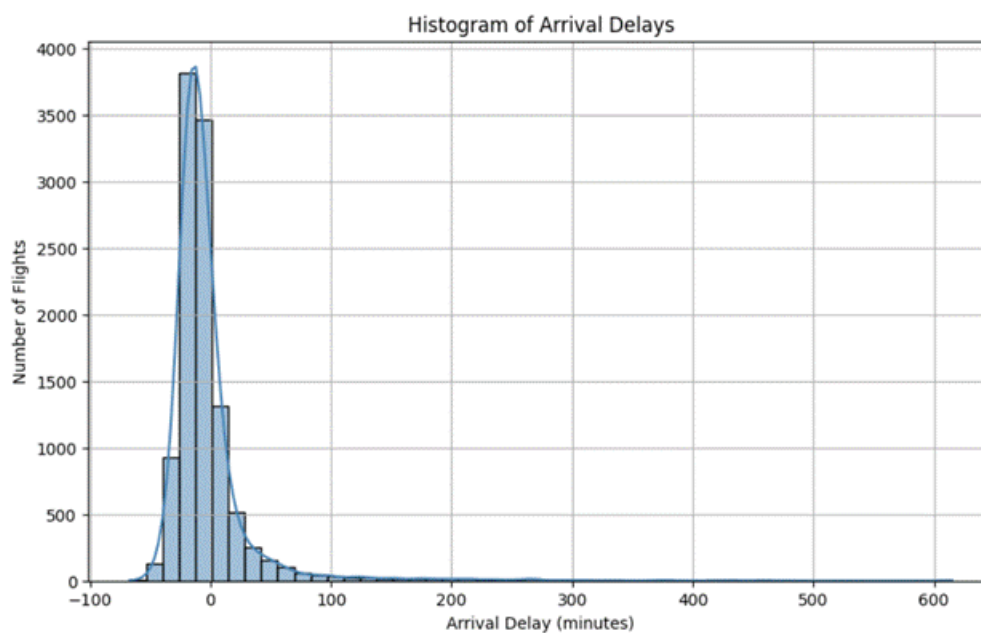
In simple words, uni variate analysis is understanding the data with single feature. Here we have displayed two different graphs such as Piechart and count plot.

Sea born package provides a wonderful function countplot. It is more useful for categorical features. With the help of countplot, we can Number of unique values in the feature.





1. The pie chart illustrates the distribution of flight delay status in the dataset. Approximately 25.4% of the flights (highlighted in orange) experienced delays, while the remaining 74.6% of flights were on time. This indicates that although a majority of flights operated as scheduled, a significant portion still faced delays, highlighting the importance of building a predictive system to identify and mitigate these occurrences.
2. From the below histogram we can say that most flights arrive on time or with minor delays, while a smaller number experience long delays. The distribution is right-skewed, indicating a few extreme delay cases.



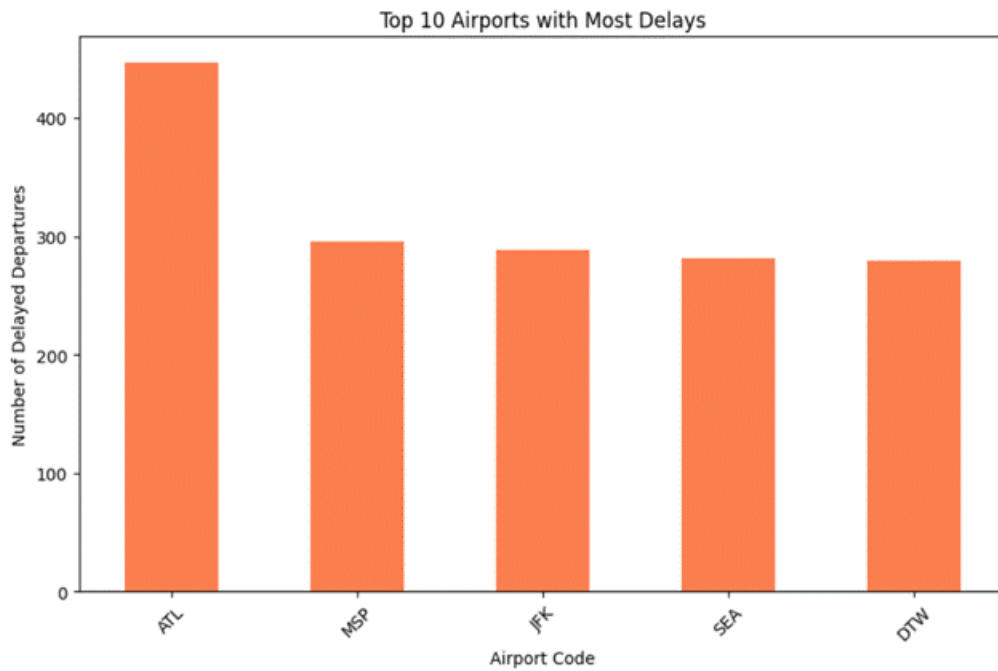
Activity 2.2: Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we can use barplot.

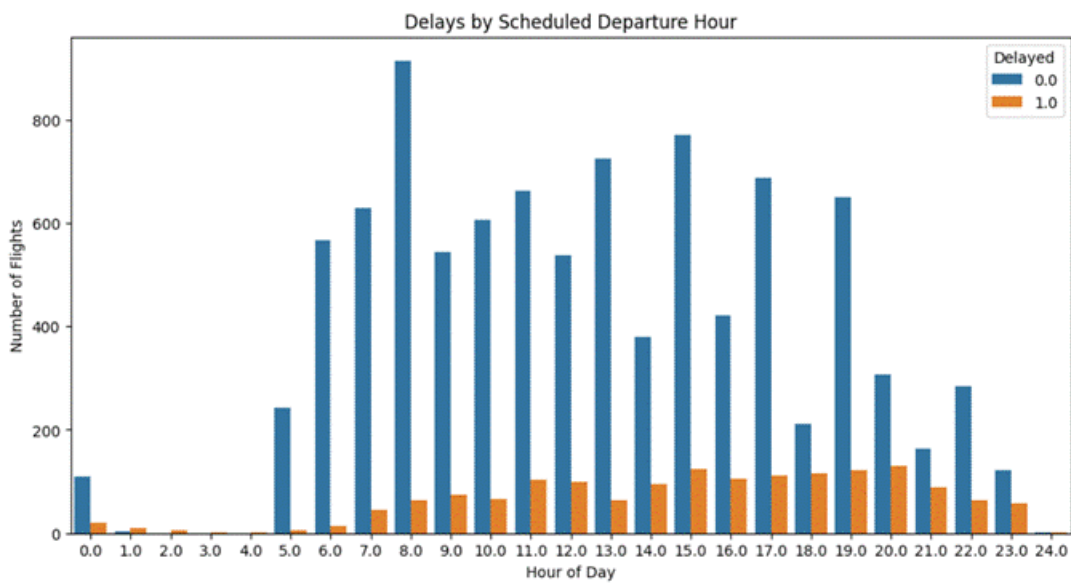
- Barplot is used here. As a 1st parameter we are passing origin and as a 2nd parameter we are

passing DEP_DEL15.

- The Most of the claim that the airport code with ATL is having more delay.



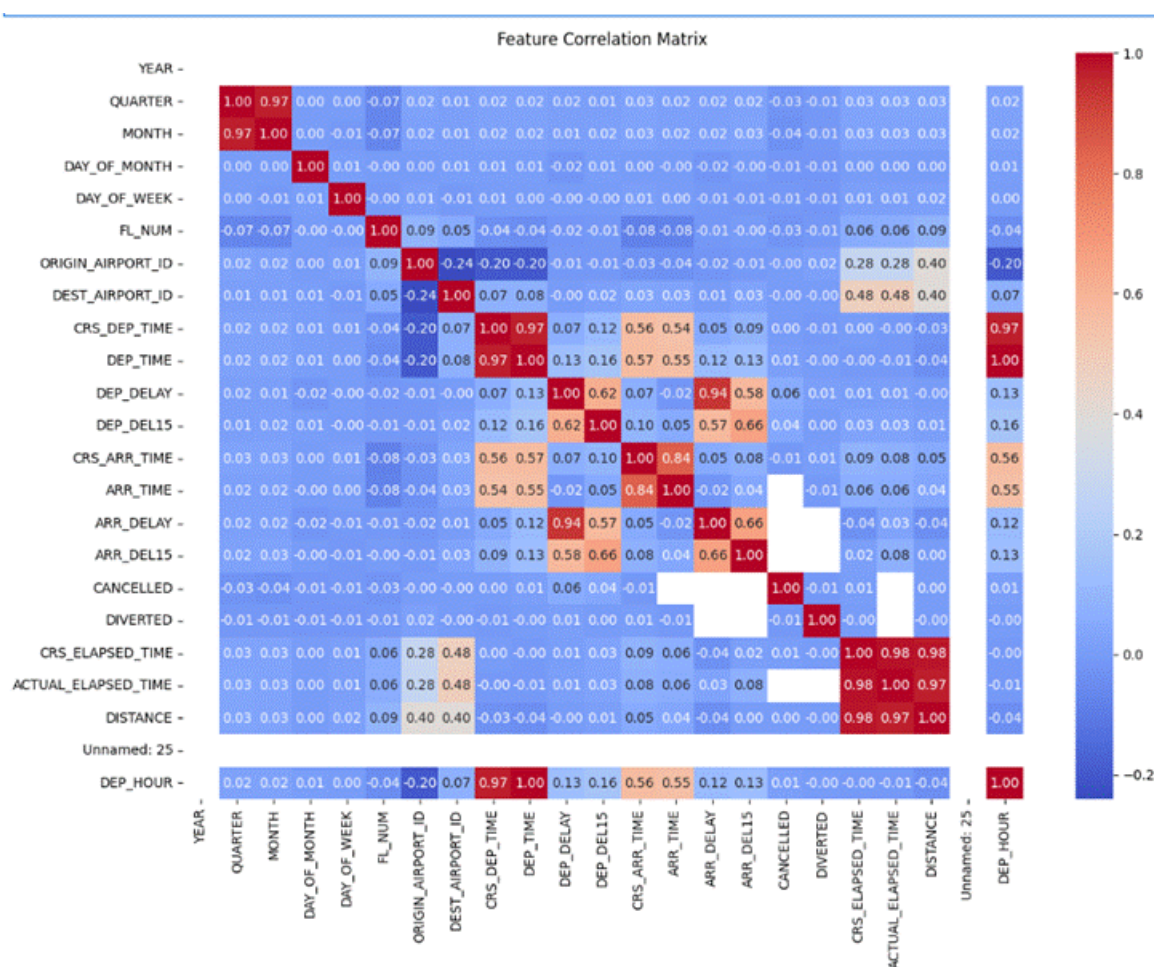
- From the below barplot we can able to figure that number of flight delays.



Activity 2.3: Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used heatmap from seaborn package.

- A correlation value close to 1 means a strong positive relationship.
- A correlation value close to -1 means a strong negative relationship.
- A value near 0 means there is little or no linear relationship.
- DEP_TIME and DEP_HOUR have a correlation of 1.00, meaning they are the same information.
- CRS_ELAPSED_TIME and ACTUAL_ELAPSED_TIME have a very high correlation of 0.98.
- DISTANCE is also strongly correlated with CRS_ELAPSED_TIME and ACTUAL_ELAPSED_TIME.
- DEP_DELAY is moderately correlated with ARR_DELAY, DEP_DEL15, and ARR_DEL15.
- Features like YEAR, MONTH, and DAY_OF_MONTH show very low correlation with most other features.
- Highly correlated features may be redundant and can be removed to avoid multicollinearity in machine learning models.



Encoding the Categorical Features:

1. The categorical Features are can't be passed directly to the Machine Learning Model. So we convert them into

Numerical data based on their order. This Technique is called Encoding.

2. Here we are importing Label Encoder from the Sklearn Library.
3. Here we are applying fit_transform to transform the categorical features to numerical features.

```
le = LabelEncoder()  
dataset['DEST'] = le.fit_transform(dataset['DEST'])  
dataset['ORIGIN'] = le.fit_transform(dataset['ORIGIN'])
```

Splitting data into train and test

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set

Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
x = dataset.iloc[:, 0:8].values  
y = dataset.iloc[:, 8:9].values  
x.shape
```

```
y = dataset.iloc[:, 5:6].values
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2, random_state = 0)
```

Handling Imbalanced dataset

1. Imbalanced data is a common problem in machine learning and data analysis, where the number of observations in one class is significantly higher or lower than the other class. Handling imbalanced data is important to ensure that the model is not biased towards the majority class and can accurately predict the minority class.
2. Here we are using SMOTE Technique.

```
import imblearn  
from imblearn.over_sampling import SMOTE
```

```
smote = SMOTE()  
x_train_smote, y_train_smote = smote.fit_resample(x_train, y_train)
```

Scaling

1. Scaling is a technique used to transform the values of a dataset to a similar scale to improve the performance of machine learning algorithms. Scaling is important because many machine learning algorithms are sensitive to the

scale of the input features.

2. Here we are using Standard Scaler.
3. This scales the data to have a mean of 0 and a standard deviation of 1. The formula is given by:
$$X_scaled = (X - X_mean) / X_std$$

```
sc = StandardScaler()  
x_train = sc.fit_transform(x_train)  
x_test = sc.transform(x_test)
```

Milestone 4: Model Building

Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying three classification algorithms. The best model is saved based on its performance.

Activity 1.1: Decision tree model

First Decision Tree is imported from sklearn Library then DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. We can find the Train and Test accuracy by X_train and X_test.

```
classifier = DecisionTreeClassifier(random_state = 0)  
classifier.fit(x_train_smote, y_train_smote)
```

Activity 1.2: Random forest model

First Random Forest Model is imported from sklearn Library then RandomForestClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. We can find the Train and Test accuracy by X_train and X_test.

```
rf = RandomForestClassifier()  
rf.fit(x_train, y_train)  
y_pred_rf = rf.predict(x_test)  
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
```

Activity 1.3: KNN model

KNN Model is imported from sklearn Library then KNeighborsClassifier algorithm is initialised and training

data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
knn = KNeighborsClassifier()
knn.fit(x_train, y_train)
y_pred_knn = knn.predict(x_test)
print("KNN Accuracy:", accuracy_score(y_test, y_pred_knn))
```

Activity 1.4: Logistic Regression model

Logistic Regression Model is imported from sklearn Library then Logistic Regression algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix is done.

```
lr = LogisticRegression()
lr.fit(x_train, y_train)
y_pred_lr = lr.predict(x_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))
```

Activity 1.5: Naïve Bayes model

Naïve Bayes Model is imported from sklearn Library then Naïve Bayes algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. We can find the Train and Test accuracy by X_train and X_test.

```
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()
nb.fit(x_train, y_train)
y_pred_nb = nb.predict(x_test)
print("Naive Bayes Accuracy:", accuracy_score(y_test, y_pred_nb))
print("Naive Bayes Report:\n", classification_report(y_test, y_pred_nb))
```

Activity 1.6: SVM model

SVM Model is imported from Sklearn Library then SVM algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new

variable. For evaluating the model, confusion matrix and classification report is done.

```
# Support Vector Machine
from sklearn.svm import SVC

svm = SVC()
svm.fit(x_train, y_train)
y_pred_svm = svm.predict(x_test)
print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))
print("SVM Report:\n", classification_report(y_test, y_pred_svm))
```

Activity 2: Testing the model

Here we have tested with Decision Tree algorithm. You can test with all algorithm. With the help of predict() function.

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Predict on test set
y_pred = rf.predict(x_test)

# Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

# Classification Report
cr = classification_report(y_test, y_pred)
print("Classification Report:\n", cr)
```

Milestone 5: Performance Testing & Hyperparameter Tuning

Activity 1: Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

Activity 1.1: Compare the model

For comparing the above four models, the compareModel function is defined.

```
# Predictions from all models (must run these first!)

y_pred_dt = classifier.predict(x_test)
y_pred_rf = rf.predict(x_test)
y_pred_knn = knn.predict(x_test)
y_pred_lr = lr.predict(x_test)
y_pred_nb = nb.predict(x_test)
y_pred_svm = svm.predict(x_test)

from sklearn.metrics import accuracy_score

model_scores = {
    "Decision Tree": accuracy_score(y_test, y_pred_dt),
    "Random Forest": accuracy_score(y_test, y_pred_rf),
    "KNN": accuracy_score(y_test, y_pred_knn),
    "Logistic Regression": accuracy_score(y_test, y_pred_lr),
    "Naive Bayes": accuracy_score(y_test, y_pred_nb),
    "SVM": accuracy_score(y_test, y_pred_svm)
}

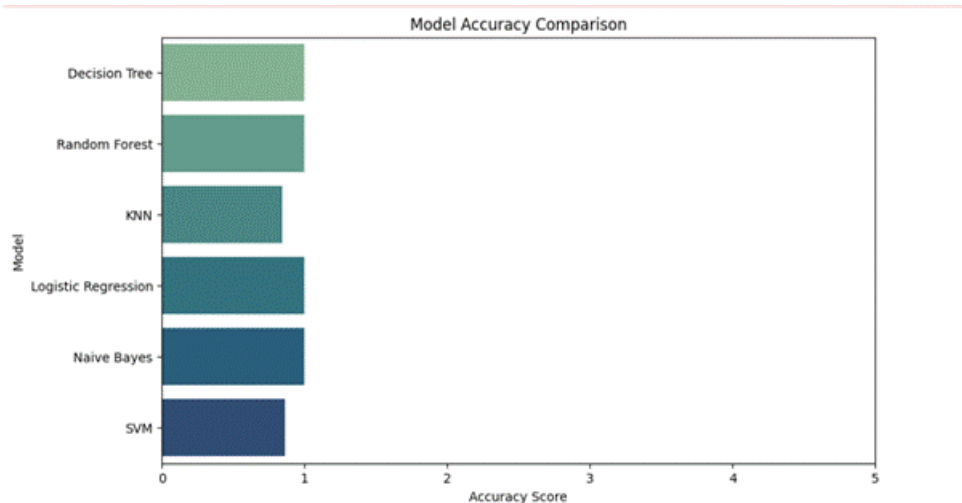
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

results_df = pd.DataFrame(list(model_scores.items()), columns=["Model", "Accuracy"])

# Print table
print(results_df)

# Plot
plt.figure(figsize=(10, 6))
sns.barplot(x="Accuracy", y="Model", data=results_df, palette="crest")
plt.title("Model Accuracy Comparison")
plt.xlabel("Accuracy Score")
plt.xlim(0, 5)
plt.show()
```

	Model	Accuracy
0	Decision Tree	1.000000
1	Random Forest	1.000000
2	KNN	0.844682
3	Logistic Regression	1.000000
4	Naive Bayes	1.000000
5	SVM	0.862483



After calling the function, the results of models are displayed as output. From the above models Decision Tree is performing well.

Milestone 6: Model Deployment

Activity 1: Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
import pickle
pickle.dump(classifier, open("flight.pkl", "wb"))
```

Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the users where he has to enter the values for predictions. The entered values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

1. Building HTML Pages
2. Building server-side script
3. Run the web application

Activity 2.1: Building Html Page:

For this project create HTML file namely

- index.html

and save them in the templates folder. Refer this [link](#) for templates.

Activity 2.2: Build Python code:

Import the libraries

```
app.py > ...  
1  from flask import Flask, render_template, request  
2  import numpy as np  
3  import pickle
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (`__name__`) as argument.

```
# Load the trained model  
model = pickle.load(open(r"C:\\Users\\yagnatharun\\flight.pkl", 'rb'))  
app = Flask(__name__)
```

Render HTML page:

```
@app.route('/')  
def home():  
    return render_template('index.html')
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered.

Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```

@app.route('/prediction', methods=['POST'])
def predict():
    # Collect numeric inputs
    flnum = int(request.form['flnum'])
    month = int(request.form['month'])
    dayofmonth = int(request.form['dayofmonth'])
    dayofweek = int(request.form['dayofweek'])
    crsarr = int(request.form['crsarr'])
    dep15 = int(request.form['dep15'])

    # One-hot encode origin
    origin = request.form['origin']
    origin1, origin2, origin3, origin4, origin5 = 0, 0, 0, 0, 0
    if origin == "dtw": origin1 = 1
    elif origin == "sea": origin2 = 1
    elif origin == "jfk": origin3 = 1
    elif origin == "atl": origin4 = 1
    elif origin == "msp": origin5 = 1

    # One-hot encode destination
    destination = request.form['destination']
    destination1, destination2, destination3, destination4, destination5 = 0, 0, 0, 0, 0
    if destination == "dtw": destination1 = 1
    elif destination == "sea": destination2 = 1
    elif destination == "jfk": destination3 = 1
    elif destination == "atl": destination4 = 1
    elif destination == "msp": destination5 = 1

    # Form the complete feature array (16 total features)
    total = [[
        flnum, month, dayofmonth, dayofweek, crsarr, dep15,
        origin1, origin2, origin3, origin4, origin5,
        destination1, destination2, destination3, destination4, destination5
    ]]

    # Predict using the model
    y_pred = model.predict(total)

    # Interpret the result
    ans = "The Flight Will be on time" if y_pred[0] == 0 else "The flight will be delayed"

    return render_template("index.html", showcase=ans)

if __name__ == '__main__':
    app.run(debug=True)

```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

```

if __name__ == '__main__':
    app.run(debug=True)

```

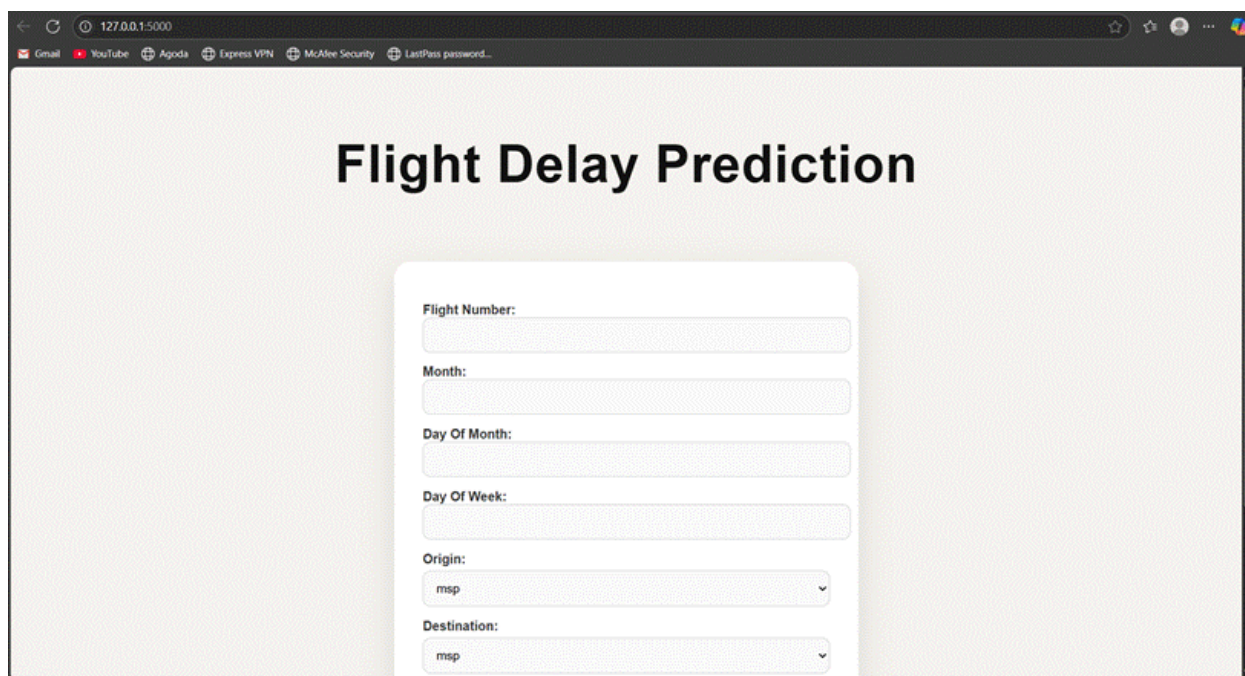
Activity 2.3: Run the web application

1. Open anaconda prompt from the start menu
2. Navigate to the folder where your python script is.

3. Now type "python app.py" command
4. Navigate to the localhost where you can view your web page.
5. Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 382-039-628
```

Now, Go the web browser and write the localhost url(<http://127.0.0.1:5000>) to get the below result



The screenshot shows a web browser window with the address bar set to 127.0.0.1:5000. The page displays a "Flight Delay Prediction" form. The form includes the following fields:

- Flight Number:
- Month:
- Day Of Month:
- Day Of Week:
- Origin:
- Destination:

The form is centered on a light gray background. The browser's address bar and tabs are visible at the top.

Flight Delay Prediction

Flight Number:

Month:

Day Of Month:

Day Of Week:

Origin:

Destination:

CRS Arrival Time (Hour 0-23):

Departed Late? (0 or 1):

Flight Delay Prediction

The flight will be delayed

Flight Delay Prediction

Flight Number:

Month:

Day Of Month:

Day Of Week:

Origin:

Destination:

CRS Arrival Time (Hour 0-23):

Departed Late? (0 or 1):

Flight Delay Prediction

The Flight Will be on time

Milestone 7: Project Demonstration & Documentation

Below mentioned deliverables to be submitted along with other deliverables

Activity 1:- Record explanation Video for project end to end solution

Activity 2:- Project Documentation-Step by step project development procedure