

# Flight Delay Prediction for aviation

Industry using  
Machine  
Learning

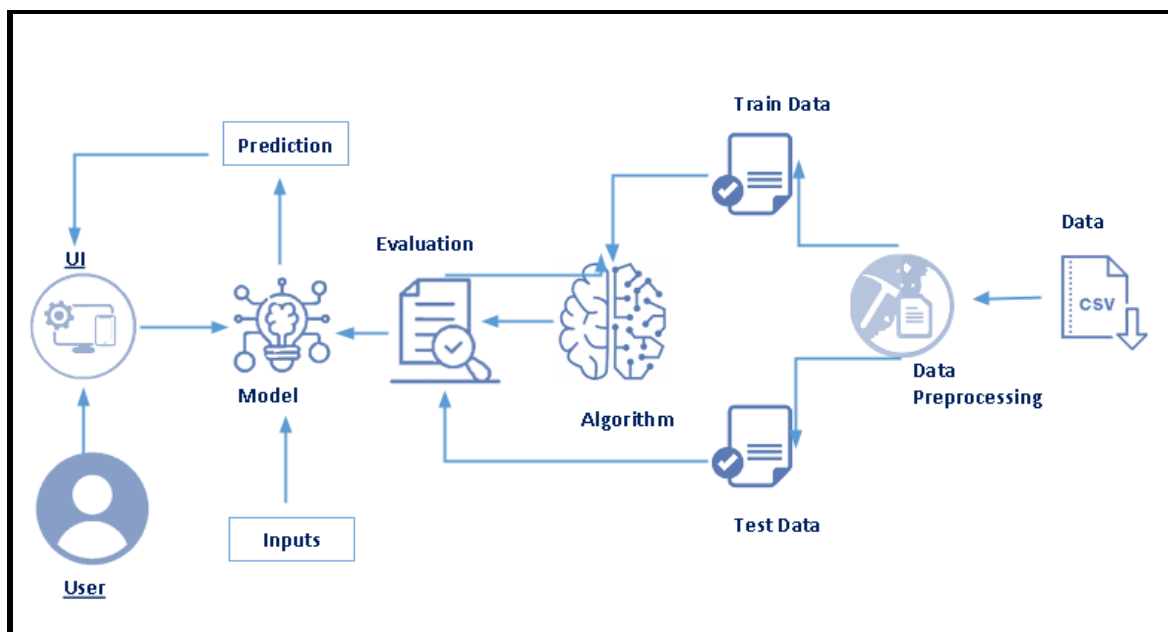
## Abstract:

OVER the last twenty years, air travel has been increasingly preferred among travelers, mainly because of its speed and in some cases comfort. This has led to phenomenal growth in air traffic and on the ground. An increase in air traffic growth has also

resulted in massive levels of aircraft delays on the ground and in the air. These delays are responsible for large economic and environmental losses. According to, taxi-out operations are responsible for 4,000 tons of hydrocarbons, 8,000 tons of nitrogen oxides and 45,000 tons of carbon monoxide emissions in the United States in 2007. Moreover, the economic impact of flight delays for domestic flights in the US is estimated to be more than \$19 Billion per year to the airlines and over \$41 Billion per year to the national economy. In response to growing concerns of fuel emissions and their negative impact on health, there is active research in the aviation industry for finding techniques to predict flight delays accurately in order to optimize flight operations and minimize delays.

Using a machine learning model, we can predict flight arrival delays. The input to our algorithm is rows of feature vector like departure date, departure delay, distance between the two airports, scheduled arrival time etc. We then use decision tree classifier to predict if the flight arrival will be delayed or not. A flight is delayed when difference between scheduled and actual arrival times is greater than 15 minutes. Furthermore, we compare decision tree classifier with logistic regression and a simple neural network for various figures of merit. Finally, it will be integrated to web based application

**Technical Architecture:**



# Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analysed 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 listed below,

- Define Problem / Problem Understanding
  - Specify the business problem
  - Business requirements
  - Literature Survey
  - Social or Business Impact.
- Data Collection & Preparation

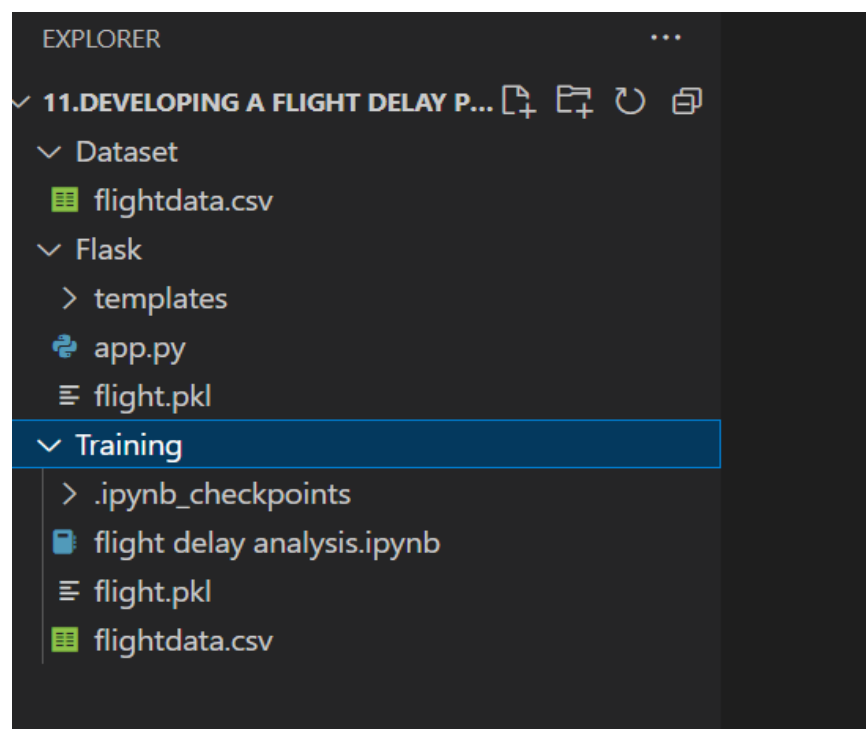
- Collect the dataset
  - Explorer Data
- Data PreparationAnalysis
  - Descriptive statistical
  - Visual Analysis
- Model Building
  - Training the model in multiple algorithms
  - Testing the model
- Performance Testing & Hyperparameter Tuning
  - Testing model with multiple evaluation metrics
  - Comparing model accuracy before & after applying hyperparameter tuning
- Model Deployment
  - Save the best model
  - Integrate with Web Framework



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

## **Project Structure:**

Create the Project folder which contains files as shown below



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- flight.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains a model training file.

## **Define Problem / Problem Understanding**

In this milestone, we go through the problem understanding

## Specify the business problem:

OVER the last twenty years, air travel has been increasingly preferred among travelers, mainly because of its speed and in some cases comfort. This has led to phenomenal growth in air traffic and on the ground. An increase in air traffic growth has also resulted in massive levels of aircraft delays on the ground and in the air. These delays are responsible for large economic and environmental losses. According to, taxi-out operations are responsible for 4,000 tons of hydrocarbons, 8,000 tons of nitrogen oxides and 45,000 tons of carbon monoxide emissions in the United States in 2007. Moreover, the economic impact of flight delays for domestic flights in the US is estimated to be more

than \$19 Billion per year to the airlines and over \$41 Billion per year to the national economy In response to growing concerns of fuel emissions and their negative impact on health, there is active research in the aviation industry for finding techniques to predict flight delays accurately in order to optimize flight operations and minimize delays.

Using a machine learning model, we can predict flight arrival delays. The input to our algorithm is rows of feature vector like departure date, departure delay, distance between the two airports, scheduled arrival time etc. We then use decision tree classifier to predict if the flight arrival will be delayed or not. A flight is delayed

when difference between scheduled and actual arrival times is greater than 15 minutes. Furthermore, we compare decision tree classifier with logistic regression and a simple neural network for various figures of merit. Finally, it will be integrated to web based application.

## Business Requirements:

To predict flight delays using machine learning, you will need to collect and process a large amount of data on past flight delays. This data should include information such as the flight's departure and arrival times, the airline, the aircraft type, and the weather conditions at the departure and arrival airports. Once you have collected and cleaned the data, you can use a

variety of machine learning techniques such as regression, decision trees, or neural networks to train a model that can predict flight delays based on this data. It is important to note that flight delay prediction is a highly complex task and requires for its resources.

### **Literature Survey:**

To predict flight delays using machine learning, you will need to collect and process a large amount of data on past flight delays. This data should include information such as the flight's departure and arrival times, the airline, the aircraft type, and the weather conditions at the departure and arrival airports.

Once you have collected and cleaned the data, you can use a variety of machine learning techniques such as regression, decision trees, or neural networks to train a model that can predict flight delays based on this data. It is important to note that flight delay prediction is a highly complex task and requires a lot of data.

The literature suggests that ML models, specifically decision tree, ANN and random forest models, have been used to predict flight delays with varying degrees of accuracy. Commonly used features include historical flight data, weather conditions, and airport operations. It also shows that a combination of data mining techniques can be used to identify the factors that contribute to flight delays.



## **Social Or Business Impact:**

The social and business impact of flight delay prediction using machine learning (ML) can be significant

From a social perspective, flight delay prediction can help improve the travel experience for passengers. By providing accurate and timely predictions of flight delays, passengers can make more informed decisions about their travel plans and potentially avoid delays or missed connections. This can lead to a reduction in travel-related stress and inconvenience.

From a business perspective, flight delay prediction can help airlines and airports improve their operations and reduce costs. By identifying and addressing the factors that contribute to flight delays, airlines and airports can take proactive measures to mitigate the impact of delays. This can lead to improved on-time performance, which can help airlines and airports attract and retain customers and increase revenue. Additionally, flight delay prediction can help airlines and

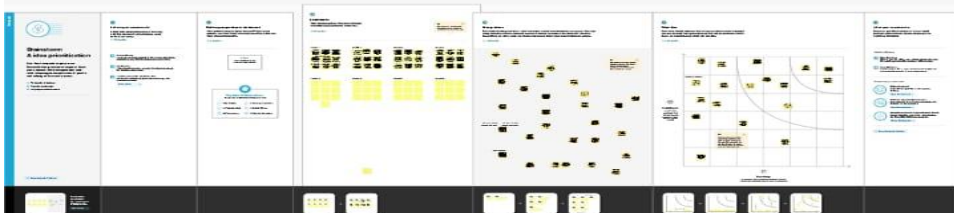
airports optimize their staffing and resource allocation, resulting in cost saving.

Empathy Map:

11:32

3.00 KB/S 4G 59%

← Untitled\_2023-04-15\_05-48-20



Brainstorm:

11:32

5.00 KB/s 49% 59%



Untitled\_2023-04-15\_05-48-20



## Advantage:

1. Improved travel planning: With the help of flight delay prediction, passengers can better plan their travel and avoid unnecessary waiting at the airport. By knowing that their flight is going to be delayed, passengers can adjust their travel plans accordingly, such as rescheduling their connecting flights or booking an alternative mode of transportation.

2. Cost savings: Flight delay prediction can help airlines optimize their flight schedules and allocate resources more efficiently, which can lead to cost savings. For example, airlines can adjust their flight schedules based on predicted delays to avoid

unnecessary fuel consumption, reduce staffing costs, and avoid penalties for delayed flights.

3.Improved safety: Delay prediction can also help improve safety by allowing airlines to proactively address

4.Customer satisfaction: By providing potential safety issues, such as bad weather, mechanical problems, or crew availability,accurate and timely information about flight delays, airlines can improve customer satisfaction and loyalty.

5.Operational efficiency: Flight delay prediction can also help airlines improve their operational efficiency by allowing them to better manage their resources, including aircraft, crew, and ground support personnel. This can lead to lower costs and higher profits for airlines.

**Disadvantage:**

1. Better planning: Flight delay prediction allows travelers to better plan their trips. By knowing in advance that a flight is likely to be delayed, passengers can adjust their schedules accordingly, avoid wasting time at the airport, and make alternative arrangements if necessary.
2. Cost savings: By avoiding flight delays, passengers can avoid additional costs such as overnight hotel stays, missed connecting flights, and lost productivity.
3. Improved safety: Delay prediction can also help improve safety by allowing airlines to proactively address potential safety issues, such as bad weather, mechanical problems, or crew availability.
4. Customer satisfaction: By providing accurate and timely information about flight delays, airlines can improve customer satisfaction and loyalty.
5. Operational efficiency: Flight delay prediction can also help airlines improve their operational efficiency by allowing them to better manage their resources, including aircraft, crew, and ground support personnel. This can lead to lower costs and higher profits for airlines.

Disadvantage of flight delay prediction



While there are many advantages to flight delay prediction, there are also some potential disadvantages:

1. False alarms: Predicting a delay that does not actually occur can create unnecessary stress and inconvenience for travelers, who may make alternate plans or rush to the airport unnecessarily.
2. Over-reliance: Some travelers may become over-reliant on delay predictions and fail to arrive at the airport early enough, leading to missed flights.
3. Limited accuracy: While delay prediction algorithms have improved significantly in recent years, they are not 100% accurate and can sometimes miss unexpected events that cause delays.
4. Technical difficulties: Technical issues such as system crashes or data errors can sometimes lead to inaccurate delay predictions.
5. Cost: The development and implementation of advanced delay prediction systems can be costly for airlines and airports, and these costs may be passed on to passengers in the form of higher ticket prices or additional fees.

Overall, while the benefits of flight delay prediction can be significant, it is important for travelers to keep in mind that predictions are not always perfect and to always arrive at the airport with enough time to catch their flight.

## Result:

1. Better planning: Flight delay prediction allows travelers to better plan their trips. By knowing in advance that a flight is likely to be delayed, passengers can adjust their schedules accordingly, avoid wasting time at the airport, and make alternative arrangements if necessary.
2. Cost savings: By avoiding flight delays, passengers can avoid additional costs such as overnight hotel stays, missed connecting flights, and lost productivity.
3. Improved safety: Delay prediction can also help improve safety by allowing airlines to proactively address potential safety issues, such as bad weather, mechanical problems, or crew availability.
4. Customer satisfaction: By providing accurate and timely information about flight delays, airlines can improve customer satisfaction and loyalty.
5. Operational efficiency: Flight delay prediction can also help airlines improve their operational efficiency by allowing them to better manage their resources,

including aircraft, crew, and ground support personnel. This can lead to lower costs and higher profits for airlines.

## Application:

Building a flight delay prediction application can be a complex task, but it can be achieved using machine learning techniques. Here are some high-level steps you can follow:

1. Gather data: You need to collect historical data on flights, including information such as origin, destination, departure

time, arrival time, and whether or not the flight was delayed. You can find this data from various sources, including government agencies and airlines themselves.

2. Clean and preprocess data: Data preprocessing is an essential step in any machine learning project. You need to clean your data by removing any missing or incorrect values, dealing with outliers, and transforming features into a suitable format for analysis.
3. Feature engineering: Feature engineering involves selecting the most important variables that can help predict flight delays. These may include weather data, time of day, airline, and the number of previous delays for a particular flight.
4. Build a model: Once you have your data ready, you can train a machine learning model using various algorithms, such as

logistic regression, decision trees, random forests, or neural networks.

5. Test and validate the model: To evaluate the performance of your model, you need to split your data into training and testing sets. You can then use metrics such as accuracy, precision, recall, and F1 score to determine how well your model is performing.
6. Deploy the model: Once you have validated your model's performance, you can deploy it as an application, either as a web or mobile app, that users can access and get real-time predictions on flight delays.

Overall, building a flight delay prediction application requires a strong understanding of data science and machine learning

techniques. It is also essential to have a reliable source of data and a robust infrastructure to handle real-time requests.

## 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.

## 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.

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

#Task 1:Problem understanding

##1)Specify business problem

##2)Business Requirement

##3)Literature survey

##4)Social/Business impact

#Task 2:Data understanding

##1)Data collection

##2>Loading Data

#Task 3:EDA

##1)Data Cleaning

##2)Data Manipulation

##3)Visualiration

#Task 4:Model building

#Task 5:Testing the model



#Task 6:Deployment  
#Task 7:Doc

Task 1:Problem understanding

1)Specify business problem

2)Business Requirement

3)Literature survey

4)Social/Business impact

Task 2:Data understanding

1)Data collection

2)Loading Data

Task 3:EDA

1)Data Cleaning

2)Data Manipulation

3)Visualiration

Task 4:Model building

Task 5:Testing the model

Task 6:Deployment

Task 7:Doc

---

# Importing The Libraries

---

```
# Importing required lib  
  
import numpy as np  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
import warnings  
warnings.filterwarnings('ignore');
```

```
# Checking for available styles

plt.style.available

# Applying styles to notebook

plt.style.use('fivethirtyeight')
```

## **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.

```
# Reading csv data  
df=pd.read_csv('/content/flightdata.csv')  
df.head()
```

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

Python

```
dataset.head()
```

Python

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	TAIL_NUM	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	...	CRS_ARR_TIME	ARR_TIME	ARR_DELAY	ARR_DEL15	CANCELLED	DIVERTED	CRS_ELAPSED_TIME	ACTUAL_ELAPSED_TIME	DISTANCE	Unnamed: 25
0	2016	1	1	1	5	DL	N836DN	1399	10397	ATL	..	2143	2102.0	-41.0	0.0	0.0	0.0	338.0	295.0	2182.0	NaN
1	2016	1	1	1	5	DL	N964DN	1476	11433	DTW	..	1435	1439.0	4.0	0.0	0.0	0.0	110.0	115.0	528.0	NaN
2	2016	1	1	1	5	DL	N813DN	1597	10397	ATL	..	1215	1142.0	-33.0	0.0	0.0	0.0	335.0	300.0	2182.0	NaN
3	2016	1	1	1	5	DL	N587NW	1768	14747	SEA	..	1335	1345.0	10.0	0.0	0.0	0.0	196.0	205.0	1399.0	NaN
4	2016	1	1	1	5	DL	N836DN	1823	14747	SEA	..	607	615.0	8.0	0.0	0.0	0.0	247.0	259.0	1927.0	NaN

rows x 26 columns

## 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.

- Handling missing values

- Handling categorical data

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.

## Handling Missing Values

- Let's find the shape of our dataset first. To find the shape of our data, the `df.shape` method is used. To find the data type, `df.info()` function is used.



```
# Checking data type
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 11231 entries, 0 to 11230
```

```
Data columns (total 26 columns):
```

#	Column	Non-Null Count	Dtype
---	-----	-----	-----
0	YEAR	11231 non-null	int64
1	QUARTER	11231 non-null	int64
2	MONTH	11231 non-null	int64
3	DAY_OF_MONTH	11231 non-null	int64
4	DAY_OF_WEEK	11231 non-null	int64
5	UNIQUE_CARRIER	11231 non-null	object
6	TAIL_NUM	11231 non-null	object
7	FL_NUM	11231 non-null	int64

8	ORIGIN_AIRPORT_ID	11231	non-null	int64
9	ORIGIN	11231	non-null	object
10	DEST_AIRPORT_ID	11231	non-null	int64
11	DEST	11231	non-null	object
12	CRS_DEP_TIME	11231	non-null	int64
13	DEP_TIME	11124	non-null	float64
14	DEP_DELAY	11124	non-null	float64
15	DEP_DEL15	11124	non-null	float64
16	CRS_ARR_TIME	11231	non-null	int64
17	ARR_TIME	11116	non-null	float64
18	ARR_DELAY	11043	non-null	float64
19	ARR_DEL15	11043	non-null	float64
20	CANCELLED	11231	non-null	float64
21	DIVERTED	11231	non-null	float64
22	CRS_ELAPSED_TIME	11231	non-null	float64
23	ACTUAL_ELAPSED_TIME	11043	non-null	float64
24	DISTANCE	11231	non-null	float64
25	Unnamed: 25	0	non-null	float64

dtypes: float64(12), int64(10), object(4)

```
"""
```

```
Types of Analysis
```

```
1)Univariate analysis
```

```
2)Bivariate analysis
```

```
3)Multivariate analysis
```

```
4)Descriptive analysis/statistics
```

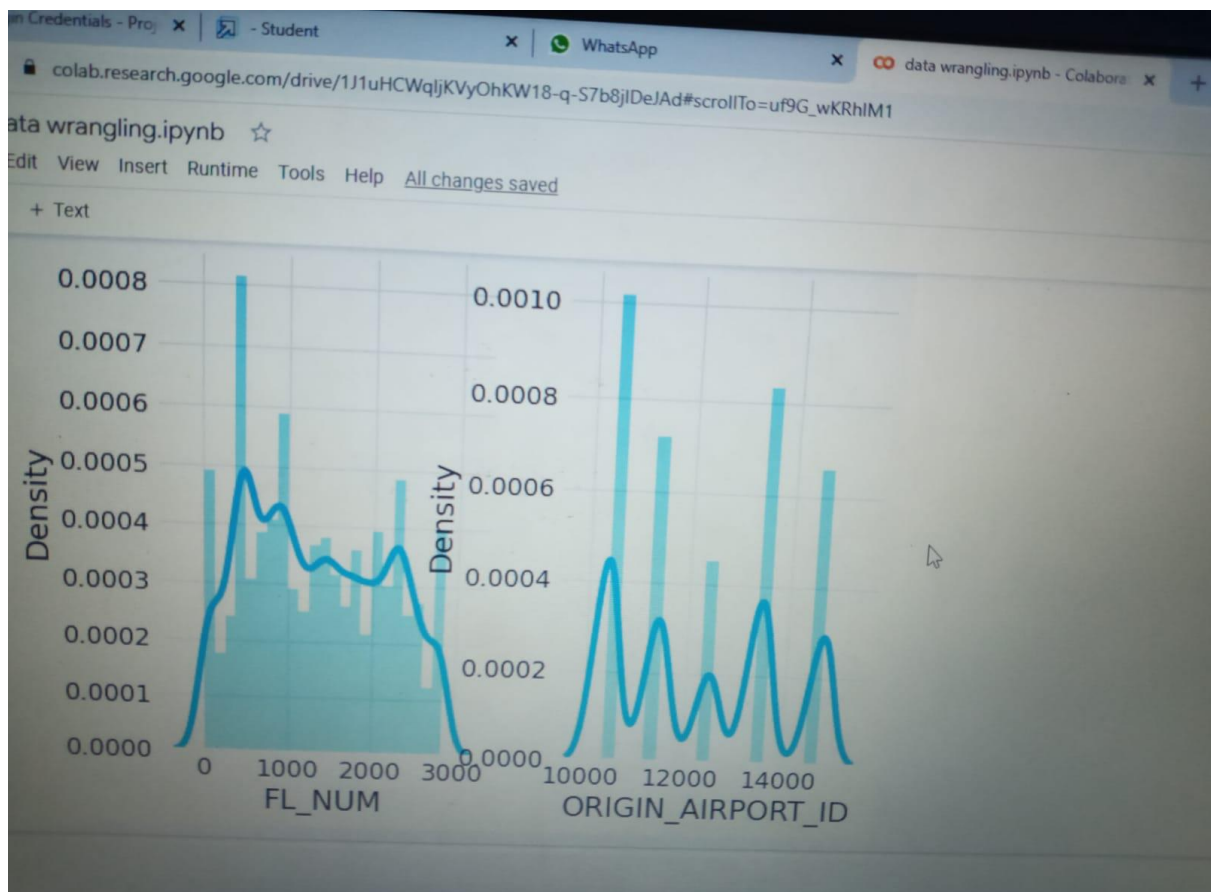
```
"""
```

```
\nTypes of Analysis\n1)Univariate
```

```
analysis\n2)Bivariate analysis\n3)Multivariate
```

```
analysis\n4)Descriptive analysis/statistics\n
```

```
#Univariate analysis -  
    Extracting info from a single column  
  
#Checking data distribution  
  
plt.subplot(121)  
  
sns.distplot(df['FL_NUM'])  
plt.subplot(122)  
sns.distplot(df['ORIGIN_AIRPORT_ID'])  
  
<Axes: xlabel='ORIGIN_AIRPORT_ID',  
ylabel='Density'>
```



```
#Creating dummy dataframe for categorical values  
df_cat=df.select_dtypes(include='object')  
df_cat.head()
```

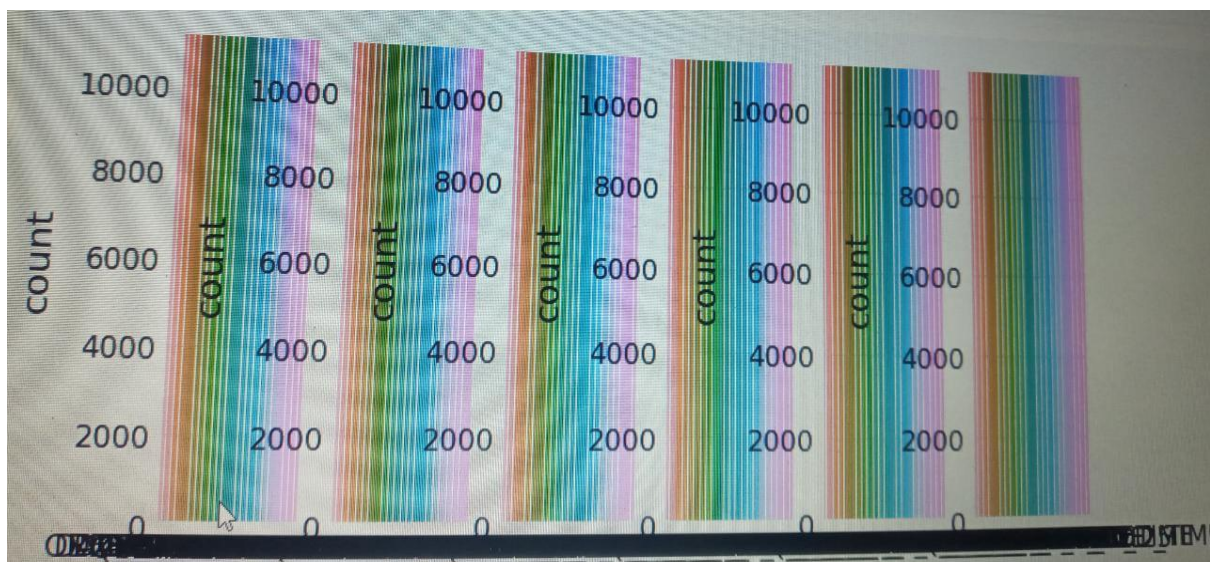


	UNIQUE_CARRIER	TAIL_NUM	ORIGIN	DEST
0	DL	N836DN	ATL	SEA
1	DL	N964DN	DTW	MSP
2	DL	N813DN	ATL	SEA
3	DL	N587NW	SEA	MSP
4	DL	N836DN	SEA	DTW

```
[ ] plt.figure(figsize=(18,4))  
for i in range(5):
```

```
plt.figure(figsize=(18,4))
for i,j in enumerate('df_cat'):
    plt.subplot(1,14,i+1)
    sns.countplot(df)
```





```
df['FL_NUM'].max()
```

2853

```
df['FL_NUM_']=[ '2.0-50.0' if x<=50.0 else "50.0-  
65.0" if x>50.0 and x<=65.0 else '65.0+' for x in df['FL_NUM']]
```

```
df.head()
```

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	TAIL_NUM	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	...	ARR_TIME	ARR_DELAY	ARR_DEL15	CANCELLED
0	2016	1	1	1	5	DL	N836DN	1399	10397	ATL	...	2102.0	-41.0	0.0	0
1	2016	1	1	1	5	DL	N964DN	1476	11433	DTW	...	1439.0	4.0	0.0	0
2	2016	1	1	1	5	DL	N813DN	1597	10397	ATL	...	1142.0	-33.0	0.0	0
3	2016	1	1	1	5	DL	N587NW	1768	14747	SEA	...	1345.0	10.0	0.0	0
4	2016	1	1	1	5	DL	N836DN	1823	14747	SEA	...	615.0	8.0	0.0	0

5 rows x 27 columns

## Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using manual encoding with the help of list comprehension..

```
df.head()
```

```
#splitting dep and indep variables
```

```
x=df.drop('FL_NUM',axis=1)
x.head()
```

```
y=df['FL_NUM']
y
```

```
0  1399
1  1476
2  1597
3  1768
```

```
4 1823 ... 11226 1715 11227 1770 11228 1823
11229 1901 11230 2005
```

```
Name: FL_NUM, Length: 11231, dtype: int64
```

## Milestone 3:

## 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.

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	TAIL_NUM	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	...	ARR_T
0	2016	1	1	1	5	DL	N836DN	1399	10397	ATL	...	21
1	2016	1	1	1	5	DL	N964DN	1476	11433	DTW	...	14
2	2016	1	1	1	5	DL	N813DN	1597	10397	ATL	...	11
3	2016	1	1	1	5	DL	N587NW	1768	14747	SEA	...	13
4	2016	1	1	1	5	DL	N836DN	1823	14747	SEA	...	6

5 rows x 27 columns

```
# importing required lib
import numpy as np
import pandas as pd
import seaborn as sns
```



```
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

**#Reading CSV data**

```
df=pd.read_csv('/content/flightdata.csv')
df.head()
```

## **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.

```
# Descriptive analysis- descriptive stat  
  
df.describe()
```

# 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 decision

# Univariate Analysis

In simple words, univariate analysis is understanding the data with a single feature. Here we have displayed two different graphs such as distplot and countplot.

The Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

# **Multivariate Analysis**

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

## **Milestone 4:**

### **Model Building**

In this milestone, we will see the model building.

## **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 four classification algorithms. The best model is saved based on its performance.

## Decision Tree Model

A function named `decisionTree` is created and train and test data are passed as the parameters. Inside the function, `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. For evaluating the model, a confusion matrix and classification report is done.

We are going to use `x_train` and `y_train` obtained above in `train_test_split` section to train our **Decision Tree Classifier** model. We're using the `fit` method and

passing the parameters as shown below.



```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(random_state = 0)
classifier.fit(x_train,y_train)
```

3]

```
DecisionTreeClassifier(random_state=0)
```

✓

```
decisiontree = classifier.predict(x_test)
```

4]

## Random Forest Model

A function named Random Forest is created and train and test data are passed as the parameters. Inside the function, Random Forest Classifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with. predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
from sklearn.ensemble import RandomForestClassifier  
rfc = RandomForestClassifier(n_estimators=10,criterion='entropy')
```

4]

```
rfc.fit(x_train,y_train)
```

5]

<ipython-input-125-b87bb2ba9825>:1: DataConversionWarning: A column-vector y was passed when you used fit: a 1D array was expected for output, but scalar y was supplied.  
ravel().

```
rfc.fit(x_train,y_train)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

6]

```
y_predict = rfc.predict(x_test)
```

## **ANN Model**

Building and training an Artificial Neural Network (ANN) using the Keras library with TensorFlow as the backend. The ANN is initialised as an instance of the Sequential class, which is a linear stack of layers. Then, the input layer and two hidden layers are added to the model using the Dense class, where the number of units and activation function are specified. The output layer is also added using the Dense class with a sigmoid activation function. The model is then compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy metric. Finally, the model is fit to the training

data with a batch size of 100, 20% validation split, and 100 epochs.

```
# Importing the Keras libraries and packages
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
# Creating ANN skleton view
```

```
classification = Sequential()
classification.add(Dense(30,activation='relu'))
classification.add(Dense(128,activation='relu'))
classification.add(Dense(64,activation='relu'))
classification.add(Dense(32,activation='relu'))
classification.add(Dense(1,activation='sigmoid'))
```

```
# Compiling the ANN model
```

```
classification.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
Epoch 99/100 1797/1797 [=====] - 4s 2ms/step - loss: 0.0586 - accuracy: 0.9789 - val_loss: 1.1199 - val_accuracy: 0.8676 Epoch 100/100 1797/1797  
[=====] - 5s 3ms/step - loss: 0.0517 - accuracy: 0.9811 - val_loss: 1.1271 - val_accuracy: 0.8648  
  
<tensorflow.python.keras.callbacks.History at 0x22721bdb7c0>
```

## Test The Model

```
## Decision tree
```

```
y_pred = classifier.predict([[129,99,1,0,0,1,0,1,1,1,0,1,1,1,1]])
```

```
print(y_pred)  
(y_pred)
```

```
[0.]
```

```
array([0.])
```

```
## RandomForest
```

```
y_pred = rfc.predict([[129,99,1,0,0,1,0,1,1,1,0,1,1,1,1]])
```

```
print(y_pred)  
(y_pred)
```

```
[0.]
```

```
array([0.])
```



In ANN we first have to save the model to the test the inputs

```
classification.save('flight.h5')
```

```
# Testing the model
```

```
y_pred = classification.predict(x_test)
```

```
y_pred
```

```
array([[3.1306639e-01],  
       [4.3961532e-19],
```



```
y_pred = (y_pred > 0.5)
y_pred
```

```
66]
```

```
.. array([[False],
          [False],
          [False],
          ...,
          [False],
          [False],
          [ True]])
```

This code defines a function named "predict\_exit" which takes in a sample\_value as an input. The function then converts the input sample\_value from a list to a numpy array. It reshapes the sample\_value array as it contains only one record. Then, it applies feature scaling to the reshaped sample\_value array using a scaler object 'sc' that should have been previously defined and fitted. Finally, the function returns the prediction of the classifier on the scaled sample\_value.

```

def predict_exit(sample_value):

    # Convert list to numpy array
    sample_value = np.array(sample_value)

    # Reshape because sample_value contains only 1 record
    sample_value = sample_value.reshape(1, -1)

    # Feature Scaling
    sample_value = sc.transform(sample_value)

    return classifier.predict(sample_value)

```

```

test=classification.predict([[1,1,121.000000,36.0,0,0,1,0,1,1,1,1,1,1,1]])
if test==1:
    print('Prediction: Chance of delay')
else:
    print('Prediction: No chance of delay.')

```

Milestone 5:

## **Performance Testing & Hyperparameter Tuning**

In this milestone, we will go through the performance testing and hyperparameter tuning.

## **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

## **Compare The Model**

For comparing the above three models

```
from sklearn import model_selection
from sklearn.neural_network import MLPClassifier
```

```

dfs = []
models = [
    ('RF', RandomForestClassifier()),
    ('DecisionTree', DecisionTreeClassifier()),
    ('ANN', MLPClassifier())
]
results = []
names = []
scoring = ['accuracy', 'precision_weighted', 'recall_weighted', 'f1_weighted', 'roc_auc']
target_names = ['no delay', 'delay']
for name, model in models:
    kfold = model_selection.KFold(n_splits=5, shuffle=True, random_state=90210)
    cv_results = model_selection.cross_validate(model, x_train, y_train, cv=kfold, scoring=scoring)
    clf = model.fit(x_train, y_train)
    y_pred = clf.predict(x_test)
    print(name)
    print(classification_report(y_test, y_pred, target_names=target_names))

```



RF

	precision	recall	f1-score	support
no delay	0.93	0.96	0.95	1936
delay	0.72	0.58	0.64	311
accuracy			0.91	2247
macro avg	0.82	0.77	0.79	2247
weighted avg	0.90	0.91	0.91	2247

DecisionTree

	precision	recall	f1-score	support
no delay	0.93	0.96	0.95	1936
delay	0.72	0.58	0.64	311
accuracy			0.91	2247
macro avg	0.82	0.77	0.79	2247
weighted avg	0.90	0.91	0.91	2247



ANN

precision

recall

f1-score

support

no delay

0.93

0.96

0.95

1936

delay

0.70

0.58

0.63

311

accuracy

0.91

2247

macro avg

0.82

0.77

0.79

2247





```
# RandomForest Accuracy
```

```
print('Training accuracy: ',accuracy_score(y_train,y_predict_train))
```

```
print('Testing accuracy: ',accuracy_score(y_test,y_predict))
```

```
Training accuracy:  0.9892030276046304
```

```
Testing accuracy:  0.89942145082332
```

```
# Making the Confusion Matrix
```

```
from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(y_test, y_predict)
```



```
# Accuracy score of desicionTree
```

```
from sklearn.metrics import accuracy_score  
desacc = accuracy_score(y_test,decisiontree)
```

```
desacc
```

```
0.8673787271918113
```

```
from sklearn.metrics import confusion_matrix
```



```
# Calculate the Accuracy of ANN
from sklearn.metrics import accuracy_score, classification_report
score = accuracy_score(y_pred, y_test)
print('The accuracy for ANN model is: {}'.format(score*100))
```

The accuracy for ANN model is: 87.2719181130396%

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

## **Comparing Model Accuracy Before & After Applying Hyperparameter Tuning**

Evaluating performance of the model From sklearn, `cross_val_score` is used to evaluate the score of the model. On the parameters, we have given rf (model

name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model by `pickle.dump()`.

**Note:** To understand cross validation

```
# giving some parameters that can be used in randomized search cv
```

```
parameters = {
```

```
    'n_estimators' : [1,20,30,55,68,74,90,120,115],
```

```
    'criterion':['gini','entropy'],
```

```
    'max_features' : ["auto", "sqrt", "log2"],
```

```
    'max_depth' : [2,5,8,10], 'verbose' : [1,2,3,4,6,8,9,10]
```

```
}
```





```
bt_params
```

```
[37]
```

```
.. {'verbose': 10,  
    'n_estimators': 90,  
    'max_features': 'log2',  
    'max_depth': 10,  
    'criterion': 'entropy'}
```



```
model = RandomForestClassifier(verbose= 10, n_estimators= 120, max_features= 'log2', max_depth= 10, criterion= 'entropy')  
RCV.fit(x_train, y_train)
```



```
RandomizedSearchCV(cv=10, estimator=RandomForestClassifier(), n_iter=4,  
                  param_distributions={'criterion': ['gini', 'entropy'],  
                                       'max_depth': [2, 5, 8, 10],  
                                       'max_features': ['auto', 'sqrt',  
                                                       'log2'],  
                                       'n_estimators': [1, 20, 30, 55, 68, 74,  
                                                       90, 120, 115],  
                                       'verbose': [1, 2, 3, 4, 6, 8, 9, 10]})
```

```
y_predict_rf = RCV.predict(x_test)
```

49]

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
```

Milestone 6:

## **Model Deployment**

In this milestone, we will go through the model deployment.

## **Save The Best Model**

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. 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(RCV,open('flight.pkl','wb'))
```

## **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 uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server side script
- Run the web application

## **Building Html Pages**

For this project create one HTML files namely

- index.html

and save them in the templates folder.

## **Build Python Code**

Import the libraries

```
# importing the necessary dependencies
from flask import Flask, request, render_template
import numpy as np
import pandas as pd
import pickle
import os
```

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.

```
model = pickle.load(open('flight.pkl', 'rb'))
```

```
app = Flask(__name__)#initializing the app
```



Render HTML page...

```
@app.route('/')  
def home():  
    return render_template("index.html")  
  
@app.route('/prediction', methods = ['POST'])
```

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 home.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:

```
def predict():
    name = request.form['name']
    month = request.form['month']
    dayofmonth = request.form['dayofmonth']
    dayofweek = request.form['dayofweek']
    origin = request.form['origin']
    if(origin == "msp"):
        origin1,origin2,origin3,origin4,origin5 = 0,0,0,0,1
    if(origin == "dtw"):
        origin1,origin2,origin3,origin4,origin5 = 1,0,0,0,0
    if(origin == "jfk"):
        origin1,origin2,origin3,origin4,origin5 = 0,0,1,0,0
    if(origin == "sea"):
        origin1,origin2,origin3,origin4,origin5 = 0,1,0,0,0
    if(origin == "alt"):
```

```

destination = request.form['destination']
if(destination == "msp"):
    destination1,destination2,destination3,destination4,destination5 = 0,0,0,0,1
if(destination == "dtw"):
    destination1,destination2,destination3,destination4,destination5 = 1,0,0,0,0
if(destination == "jfk"):
    destination1,destination2,destination3,destination4,destination5 = 0,0,1,0,0
if(destination == "sea"):
    destination1,destination2,destination3,destination4,destination5 = 0,1,0,0,0
if(destination == "alt"):
    destination1,destination2,destination3,destination4,destination5 = 0,0,0,1,0
dept = request.form['dept']
arrtime = request.form['arrtime']
actdept = request.form['actdept']
dept15=int(dept)-int(actdept)
total = [[name,month,dayofmonth,dayofweek,origin1,origin2,origin3,origin4,origin5,destination1,destination2,destination3,destination4,destination5,i
#print(total)
y_pred = model.predict(total)

```

Here the route for prediction is given and necessary steps are performed in order to get the predicted output.

Main Function:

```
if name == 'main':
```

```
    app.run(debug=True)
```

## Run The Web Application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- 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" (lazy loading)

\* Environment: production

WARNING: This is a development server. Do not use it in a production deployment.  
Use a production WSGI server instead.

\* Debug mode: on

\* Running on <http://127.0.0.1:5000/> (Press CTRL+C to quit)

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

← → ↻ ⓘ 127.0.0.1:5000 🔍 ☆ 🌐 🔄 📄 👤 ⋮

# Prediction of Flight Delay

Enter the Flight Number :

Month :

Day of Month :

Day of Week :


origin  ▼

destination  ▼

Scheduled Departure Time :

Scheduled Arrival Time :

Actual Departure Time :



Input 1- Now, the user will give inputs to get the predicted result after clicking onto the submit button.

← → ↻ 127.0.0.1:5000 🔍 ☆ 🌐 🍷 📄 👤 ⋮

# Prediction of Flight Delay

Enter the Flight Number :

Month :

Day of Month :

Day of Week :


origin :

destination :

Scheduled Departure Time :

Scheduled Arrival Time :

Actual Departure Time :





← → ↻ ⓘ localhost:5000/prediction ☆ 🌐 🛠 🎵 👤 ⋮

# Prediction of Flight Delay

Enter the Flight Number :

Month :

Day of Month :

Day of Week :

origin  ▼

destination  ▼

Scheduled Departure Time :

Scheduled Arrival Time :





