# SmartInternz Externship Program : Applied Data Science

## Project title : Flight Delay Prediction using IBM Cloud

**Done By :**

1) Madhumitha Rajagopal (20BCE1912)
2) Aswin Chander N (19BCE1034)
3) Tarun R (19BCE1173)
4) Varun Shivakumar (19BCE1206)

## 1. Introduction

### 1.1 Overview :

Over the past 20 years, air travel has grown in popularity among tourists, primarily due to its efficiency and, in some cases, comfort. The result has been a remarkable increase in land traffic and air traffic. Significant levels of aircraft delays on the ground and in the air have also been brought on by an increase in air traffic.

These delays are responsible for large economic and environmental losses. These delays not only cause inconveniences to the airlines but also to the passengers. The result is an increase in travel time which increases the expenses associated with food and lodging and ultimately causes stress among passengers. The airlines are victims of extra costs associated with their crews, aircraft repositioning, fuel consumption while trying to reduce elapsed times, and many others which together tamper their reputation and often result in a loss of demand by passengers.

### 1.2 Purpose :

Given the vast amount of data on flight travels, valuable insights can be drawn from this data to allow us to gain a better understanding of flight delays. Moreover, with the abundance in data, machine learning models can be trained to possibly predict these delays, something that may prove very valuable for individual travelers and businesses alike. Hence, the main objective of this machine learning model is to predict flight delays accurately in order to optimize flight operations and minimize the delays.

## 2. Literature Survey

### 2.1 Prior Work

Chakrabarty [5] proposed a Model which made use of Gradient Boosting Classifier to predict the arrival delay for AA(American Airlines) among the top 5 busiest US airports. This paper was used to understand the basic underlying principles of how gradient boosting can be used to enhance machine learning models for classification.

Manna [8] created a model using Gradient Boosting Regressor after analyzing the raw flight data. The model aimed to predict flight arrival and departure delays. This paper was referred to understand the research on Machine Learning Algorithm Gradient Boosted Decision Tree and how it was applied to flight delay prediction.

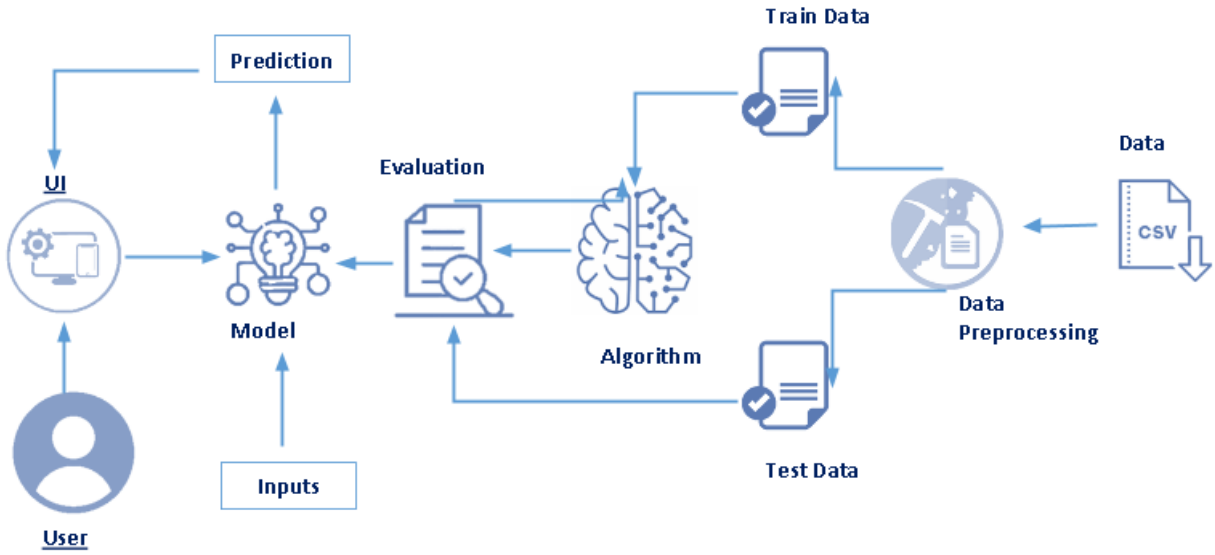
### 2.2 Proposed Solution

Our algorithm takes in rows of feature vectors that include departure date, departure delay, distance between airports, and scheduled arrival time. Using a decision tree classifier, we determine whether the flight will experience a delay, which we define as a difference greater than 15 minutes between scheduled and actual arrival times. We also compare the decision tree

classifier's performance to that of logistic regression and a basic neural network using different metrics. Finally, we utilize Flask and IBM Cloud to deploy the algorithm.

## 3. Theoretical Analytics

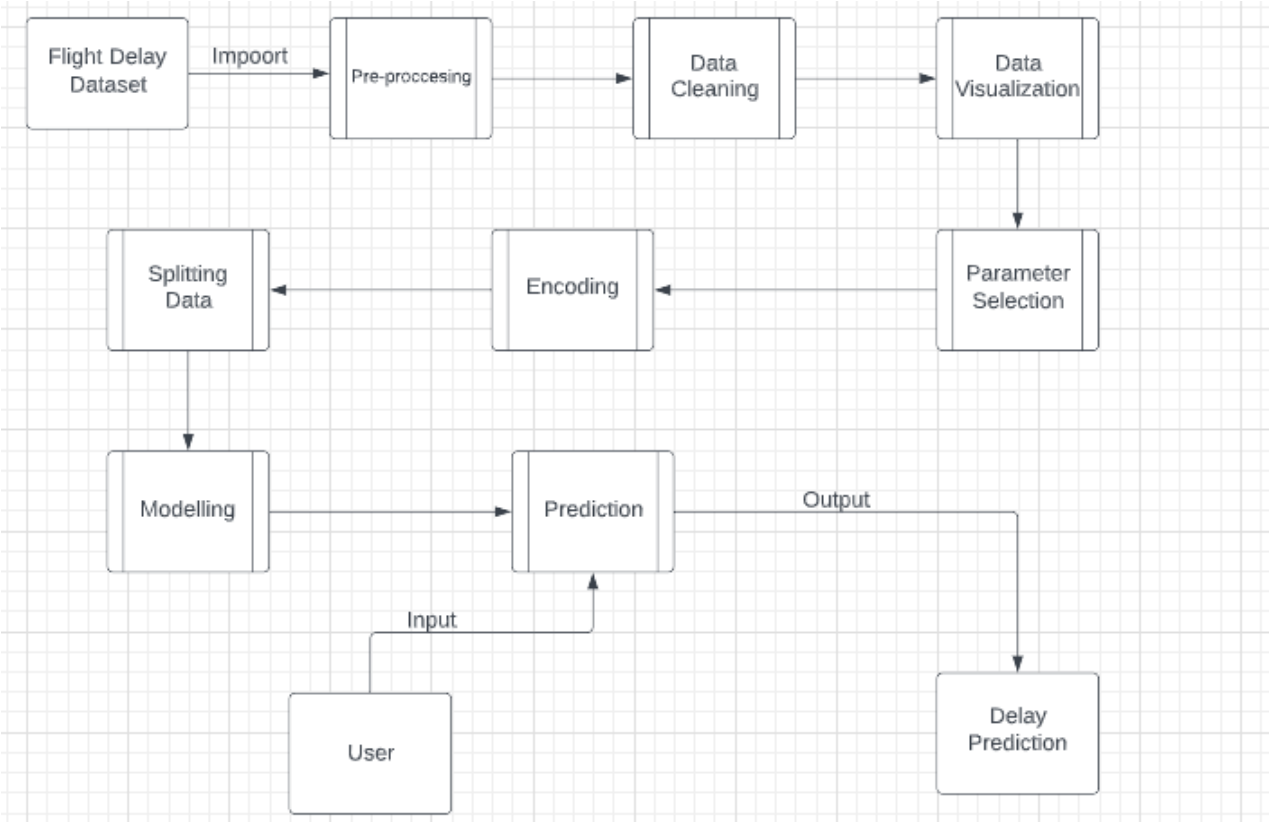
### 3.1 Block Diagram




### 3.2. Hardware and Software

- Laptop
- Anaconda Navigator
- Jupyter Notebook
- Spyder
- IBM Cloud

## 4. Flow Chart and Analysis



1. Collect the data ~ collect the data from open sources.
2. Pre- process the data ~ import the required libraries, import the dataset, take care of missing values, checking null values and splitting the data into Train and Test.
3. Analyze the data collected ~ get the summary of the data and statistical values like count, mean and standard deviation
4. Visualize the data ~ we use libraries such as Matplotlib and Seaborn. This helps us detect patterns, trends and correlations.
5. Model Building ~ Training and testing the model and evaluating the model.
6. Build the application ~ After the model is built, we integrate it to a web application so that normal users can also use it to predict the flight delay.

Supervised machine learning algorithms have been used extensively in different domains of machine learning like pattern recognition, data mining and machine translation. Similarly, there have been several attempts to apply the various supervised or unsupervised machine learning algorithms to the analysis of air traffic data.

As air travel has a significant role in the economy of agencies and airports, it is necessary for them to increase the quality of their services. One of the important modern life challenges of airports and airline agencies is flight delay. In addition, delay in flight makes passengers concerned and this matter causes extra expenses for the agency and the airport itself.

## 5. Module Explanation

### 5.1. DATA COLLECTION

The dataset used in this project is collected from Kaggle ( flightdelay.csv ). The dataset consists of approximately 11,000 data entries.

Data collection is the process of gathering and measuring information on variables of interest, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes.

During data collection, the researchers must identify the data types, the sources of data, and what methods are being used. We will soon see that there are many different data collection methods. There is heavy reliance on data collection in research, commercial, and government fields.

```
[4] dataset.columns

Index(['YEAR', 'QUARTER', 'MONTH', 'DAY_OF_MONTH', 'DAY_OF_WEEK',
       'UNIQUE_CARRIER', 'TAIL_NUM', 'FL_NUM', 'ORIGIN_AIRPORT_ID', 'ORIGIN',
       'DEST_AIRPORT_ID', 'DEST', 'CRS_DEP_TIME', 'DEP_TIME', 'DEP_DELAY',
       'DEP_DEL15', 'CRS_ARR_TIME', 'ARR_TIME', 'ARR_DELAY', 'ARR_DEL15',
       'CANCELLED', 'DIVERTED', 'CRS_ELAPSED_TIME', 'ACTUAL_ELAPSED_TIME',
       'DISTANCE', 'Unnamed: 25'],
      dtype='object')
```

Figure 3 : Column names of the dataset

## 5.2. DATA PRE-PROCESSING

Firstly we import the libraries required in order to pre-process the data.

```
[21] import sys
     import pandas as pd       #Linear Algebra
     import numpy as np        #Data Preprocessing
     import seaborn as sns     #Data Visualisation
     import pickle
     %matplotlib inline
     from sklearn.preprocessing import LabelEncoder      #Label Encoder from Sklearn
     from sklearn.preprocessing import OneHotEncoder     #One-Hot Encoding from Sklearn
     from sklearn.model_selection import train_test_split #Split data in train and test array
     from sklearn.preprocessing import StandardScaler
     from sklearn.tree import DecisionTreeClassifier      #ML Algorithm
     from sklearn.metrics import accuracy_score           #Calculate Accuracy Score
     import sklearn.metrics as metrics                    #Confusion Matrix
```

a) Numpy- It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines. Pandas objects are very much dependent on NumPy objects.

b) Pandas- It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

c) Matplotlib and Seaborn: Both are the data visualization library used for plotting graphs which will help us to understand the data.

d) Accuracy score: used in classification type problems and for finding accuracy it is used.

e) Train_test_split: used for splitting data arrays into training data and for testing data.

f) Pickle: to serialize your machine learning algorithms and save the serialized format to a file.

g) Metrics: The sklearn.metrics module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decision values.

h) Classifier: A classifier algorithm is used to map input data to a target variable through decision rules and can be used to predict and understand what characteristics are associated with a specific class or target.

i) Decision Tree Classifier: A decision tree is a tree-like structure whereby an internal node represents an attribute, a branch represents a decision rule, and the leaf nodes represent an outcome. This works by splitting the data into separate partitions according to an attribute selection measure.

j) Label Encoding: It is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering.

k) Standard Scaler: Sklearn its main scaler, the StandardScaler, uses a strict definition of standardization to standardize data. It purely centers the data by using the following formula, where u is the mean and s is the standard deviation.

l) One Hot Encoding:- One hot encoding allows the representation of categorical data to be more expressive.Many machine learning algorithms cannot work with categorical data directly. The categories must be converted into numbers. This is required for both input and output variables that are categorical.

### 5.2.1. ANALYZING DATA

In our dataset both numerical and categorical data are present, but it is not necessary that all the continuous data which we are seeing has to be continuous in nature.

a) Describe() functions are used to compute values like count, mean, and standard deviation to give a summary type of data.

```
[22] #description of the data
     dataset.describe()
```

| | YEAR | QUARTER | MONTH | DAY_OF_MONTH | DAY_OF_WEEK | FL_NUM | ORIGIN_AIRPORT_ID | DEST_AIRPORT_ID | CRS_DEP_TIME | DEP_TIME | ... | ORIGIN_ATL | O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 11231.0 | 11231.000000 | 11231.000000 | 11231.000000 | 11231.000000 | 11231.000000 | 11231.000000 | 11231.000000 | 11231.000000 | 11124.000000 | ... | 11231.000000 | 11 |
| mean | 2016.0 | 2.544475 | 6.628973 | 15.790758 | 3.960199 | 1334.325617 | 12334.516695 | 12302.274508 | 1320.798326 | 1327.189410 | ... | 0.276022 | |
| std | 0.0 | 1.090701 | 3.354678 | 8.782056 | 1.995257 | 811.875227 | 1595.026510 | 1601.988550 | 490.737845 | 500.306462 | ... | 0.447048 | |
| min | 2016.0 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 7.000000 | 10397.000000 | 10397.000000 | 10.000000 | 1.000000 | ... | 0.000000 | |
| 25% | 2016.0 | 2.000000 | 4.000000 | 8.000000 | 2.000000 | 624.000000 | 10397.000000 | 10397.000000 | 905.000000 | 905.000000 | ... | 0.000000 | |
| 50% | 2016.0 | 3.000000 | 7.000000 | 16.000000 | 4.000000 | 1267.000000 | 12478.000000 | 12478.000000 | 1320.000000 | 1324.000000 | ... | 0.000000 | |
| 75% | 2016.0 | 3.000000 | 9.000000 | 23.000000 | 6.000000 | 2032.000000 | 13487.000000 | 13487.000000 | 1735.000000 | 1739.000000 | ... | 1.000000 | |
| max | 2016.0 | 4.000000 | 12.000000 | 31.000000 | 7.000000 | 2853.000000 | 14747.000000 | 14747.000000 | 2359.000000 | 2400.000000 | ... | 1.000000 | |

8 rows × 32 columns

### 5.2.1. ANALYZING DATA

In our dataset both numerical and categorical data are present, but it is not necessary that all the continuous data which we are seeing has to be continuous in nature.

b) info() method provides the summary of the dataset.

```
[5] #information about the dataframe
    dataset.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)
memory usage: 2.2+ MB
```

### 5.2.2. IMPORTING DATASET

The dataset represents the data for the flight like a year, origin airport, flight number etc.

We load the dataset file .csv file into Pandas.

```
[2]  #Load the dataset
     dataset = pd.read_csv("flightdata.csv")
```

After we load the dataset, we check the top five records of the dataset, simply write the data frame name.head().

```
[23] #get the first n rows
     dataset.head()
```

| | YEAR | QUARTER | MONTH | DAY_OF_MONTH | DAY_OF_WEEK | UNIQUE_CARRIER | TAIL_NUM | FL_NUM | ORIGIN_AIRPORT_ID | DEST_AIRPORT_ID | ... | ORIGIN_ATL | ORIGIN_DTW | ORIGIN_JFK |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2016 | 1 | 1 | 1 | 5 | DL | N836DN | 1399 | 10397 | 14747 | ... | 1 | 0 | 0 |
| 1 | 2016 | 1 | 1 | 1 | 5 | DL | N964DN | 1476 | 11433 | 13487 | ... | 0 | 1 | 0 |
| 2 | 2016 | 1 | 1 | 1 | 5 | DL | N813DN | 1597 | 10397 | 14747 | ... | 1 | 0 | 0 |
| 3 | 2016 | 1 | 1 | 1 | 5 | DL | N587NW | 1768 | 14747 | 13487 | ... | 0 | 0 | 0 |
| 4 | 2016 | 1 | 1 | 1 | 5 | DL | N836DN | 1823 | 14747 | 11433 | ... | 0 | 0 | 0 |

5 rows × 34 columns

## 5.3. DATA VISUALISATION

Data visualization is the representation of data through use of common graphics, such as charts, plots, infographics, and even animations. These visual displays of information communicate complex data relationships and data-driven insights in a way that is easy to understand.

A) Scatter Plot

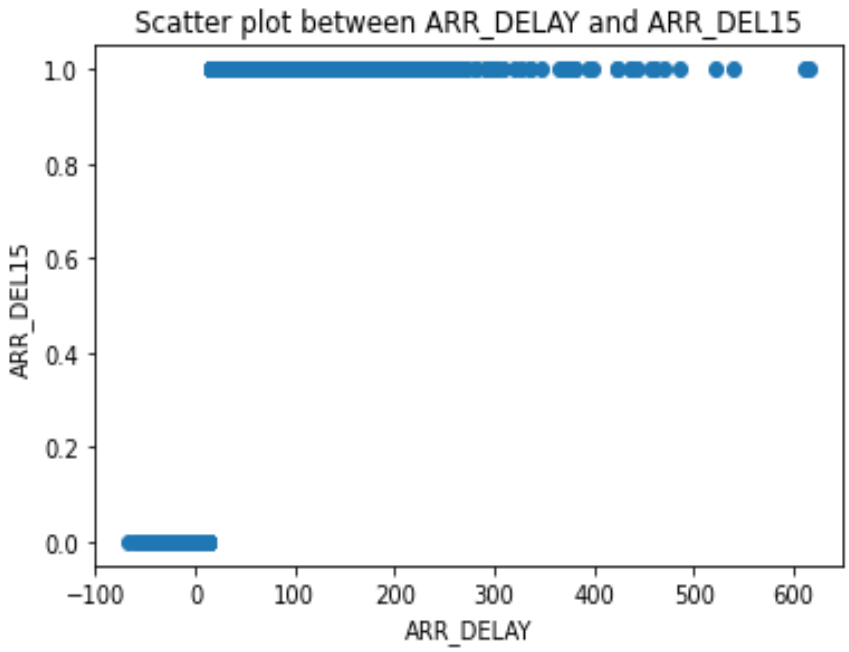



Figure 9 : Scatter Plot

A scatter plot uses dots to represent values for two different numeric variables. The position of each dot on the horizontal and vertical axis indicates values for an individual data point. Scatter plots are used to observe relationships between variables.

B) Cat Plot

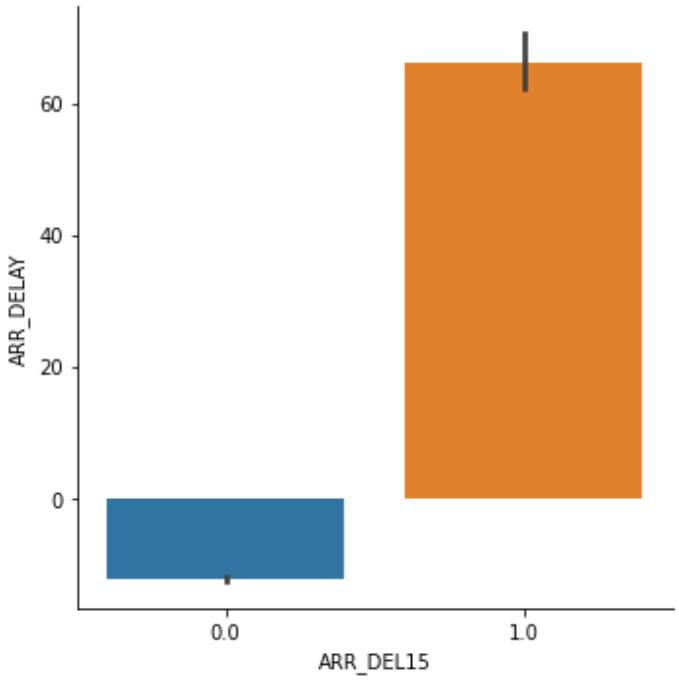



Figure 10 : Cat-plot

The new catplot function provides a new framework giving access to several types of plots that show relationships between numerical variables and one or more categorical variables, like boxplot, stripplot and so on. Catplot can handle 8 different plots currently available in Seaborn. Figure-level interface for drawing categorical plots onto a FacetGrid. This function provides access to several axes-level functions that show the relationship between a numerical and one or more categorical variables using one of several visual representations.

C) Heat Map

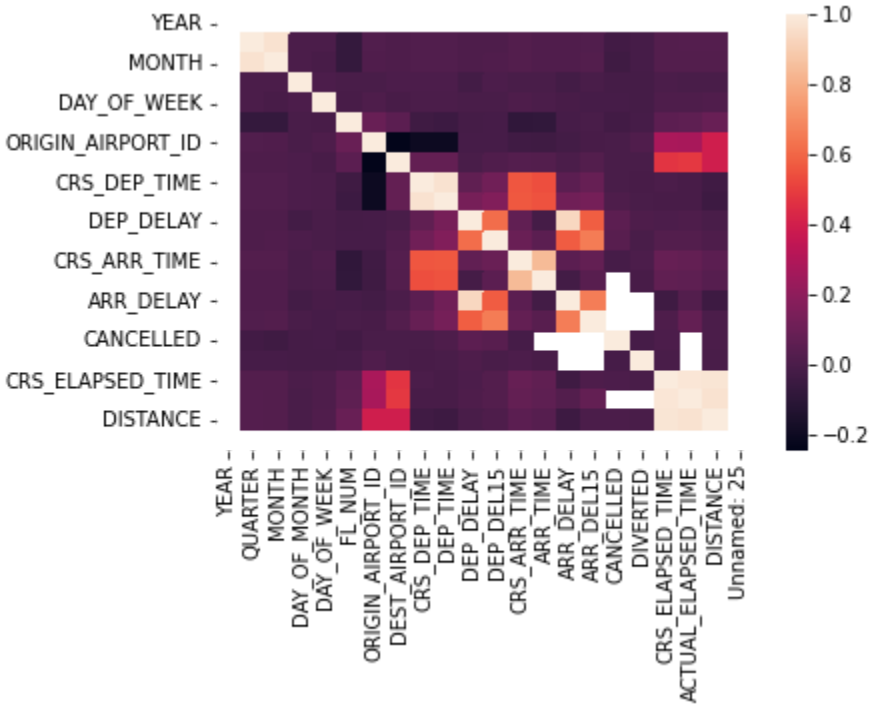



Figure 11 : Heatmap

A data visualization approach known as a heat map (or heatmap) uses color to depict a phenomenon's size in two dimensions. The reader will receive clear visual clues about how the occurrence is clustered or fluctuates over space from the fluctuation in color, which may be by hue or intensity. The cluster heat map and the spatial heat map are two fundamentally distinct types of heat maps. In a cluster heat map, magnitudes are arranged into a matrix of fixed cell size, where the rows and columns represent discrete phenomena and categories. The sorting of the rows and columns is deliberate and somewhat arbitrary, with the intention of suggesting clusters or depicting them as they are identified through statistical analysis.

## 5.4. DATA CLEANING

A structured dataset includes multiple columns with combinations of numerical as well as categorical variables. A machine can only understand the numbers. It cannot understand the text. That's essentially the case with Machine Learning algorithms too.We need to convert each text category to numbers in order for the machine to process those using mathematical equations. For this we use Label Encoding.

Label Encoding is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering. After performing label encoding alphabetical classes are converted to numeric.

Now each categorical feature with n categories is to be transformed into n binary features. The most popular way to encode nominal features is one-hot-encoding.

For this we import the OneHotEncoder class and create a new instance with the output data type set to integer. This doesn't change anything to how our data will be interpreted, but will improve the readability of our output. The output of this transformation will be a sparse matrix, this means we'll have to transform the matrix into an array (.toarray()) before we can pour it into a dataframe. You can omit this step by setting the sparse parameter to False when initiating a new class instance.

## 5.5. SPLIT THE DATASET INTO TRAIN SET AND TEST SET

The train-test split is a technique for evaluating the performance of a machine learning algorithm.

- Train Dataset: Used to fit the machine learning model.
- Test Dataset: Used to evaluate the fit machine learning mode

In this module we allocate 80% of the dataset to training set and the remaining 20% to test set.We will create 4 sets— X_train (training part of the matrix of features), X_test (test part of the matrix of features), Y_train (training part of the dependent variables associated with the X train sets, and therefore also the same indices), Y_test (test part of the dependent variables associated with the X test sets, and therefore also the same indices.

We have to make sure you split the data in a random manner. To help us with this task, the Scikit library provides a tool, called the Model Selection library. There is a class in the library which is,‘train_test_split.’ Using this we can easily split the dataset into the training and the testing datasets in various proportions.We can split our dataset into a train set and test using the train_test_split class from scikit learn library.

## 5.6. MODEL BUILDING

### 5.6.1. SMOTE TECHNIQUE

SMOTE (synthetic minority oversampling technique) algorithm is selected to process the imbalanced datasets. The SMOTE algorithm is an oversampling technology based on the KNN algorithm. It improves the simple random oversampling algorithm of randomly copying a few samples to increase the sample size, which can avoid overfitting and effectively improve the generalization ability of the model.

### 5.6.2. DECISION TREE CLASSIFIER

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. We are using this algorithm as Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of the root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

## 5.7. APPLICATION BUILDING

### 5.7.1. HTML FOR FRONT-END

The part of a website that the user interacts with directly is termed the front end. For this we use HTML to create the front end part of the web page. We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages. We make the front-end attractive and user friendly for the convenience of the user.

For pages intended to be viewed in a web browser, the HyperText Markup Language, or HTML, is the accepted markup language. Scripting languages like JavaScript and technologies like Cascading Style Sheets (CSS) can help.

HTML documents are received by web browsers from a web server or local storage and are then rendered into multimedia web pages. HTML initially provided hints for the document's look in addition to semantic descriptions of a web page's structure.

### 5.7.2. DEPLOYMENT IN FLASK

We will design a web application where the user will input all the attribute values and the data will be given to the model. Based on the training given to the model, the model will predict if the flight will be delayed or not.

Here we import the libraries, then using app=Flask(__name__) we create an instance of flask. @app.route(‘/’) is used to tell flask what URL should trigger the function index() and in the function index, we use render_template(‘index.html’) to display the script index.html in the browser.

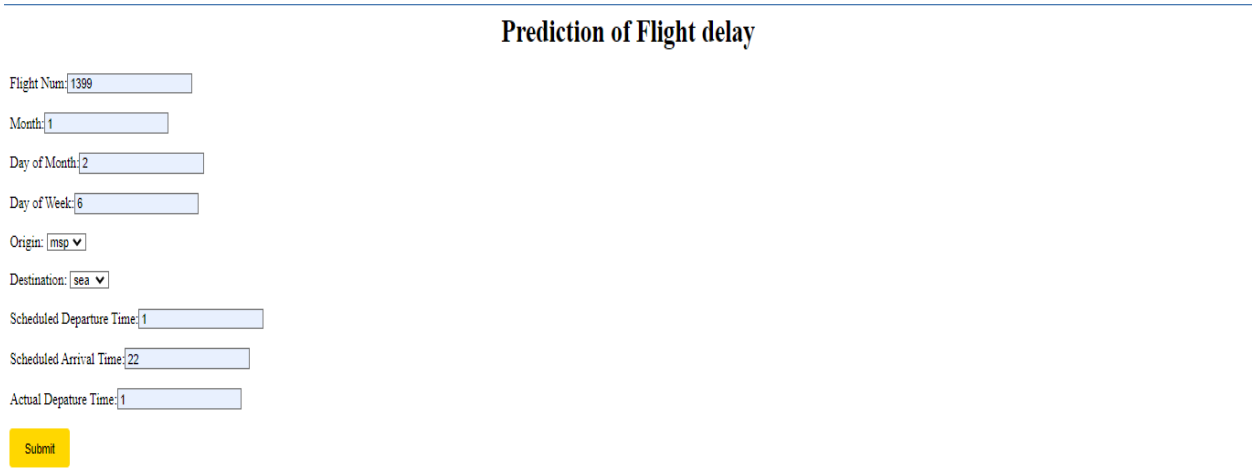
## 6. RESULT

```
In [35]: from sklearn.tree import DecisionTreeClassifier
         classifier = DecisionTreeClassifier(random_state = 0)
         classifier.fit(x_train_smote,y_train_smote)

Out[35]: DecisionTreeClassifier
         DecisionTreeClassifier(random_state=0)

In [36]: decisiontree = classifier.predict(x_test)

In [37]: from sklearn.metrics import accuracy_score
         acc = accuracy_score(y_test,decisiontree)
         acc

Out[37]: 1.0

In [38]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test,decisiontree)

In [39]: cm

Out[39]: array([[1938,    0],
                [   0,  309]], dtype=int64)

In [40]: import pickle
         pickle.dump(classifier,open("flight.pkl","wb"))
```

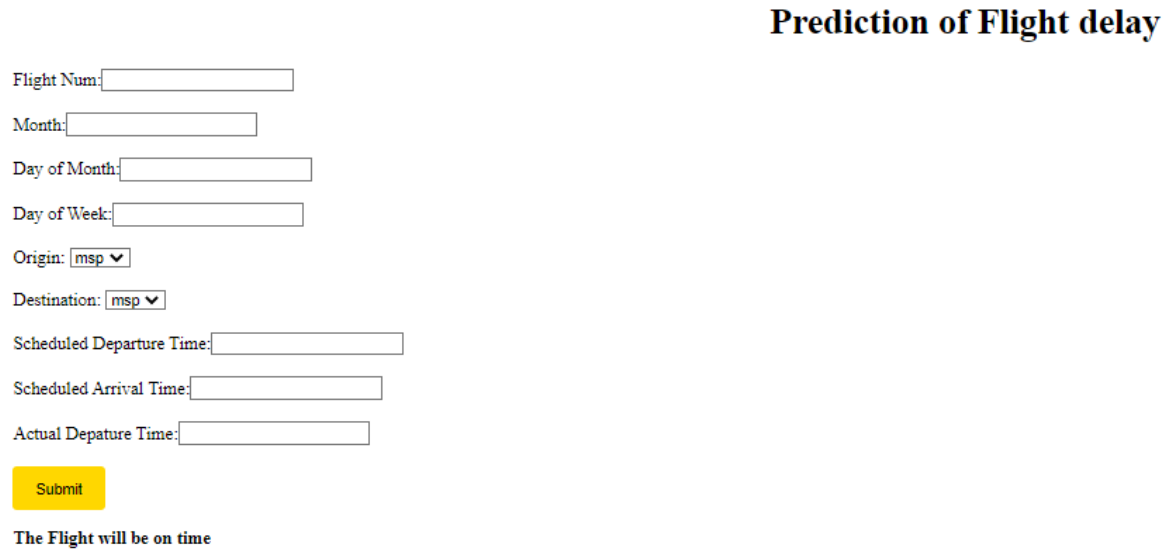
Here we have done the Decision Tree Model and have connected it to the flask application through the flight.pkl file.

```
from flask import Flask,render_template,request
import numpy as np
import pandas as pd
import pickle
import os

model = pickle.load(open('flight.pkl','rb'))
app = Flask(__name__)

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


@app.route('/prediction',methods = ['POST'])
```

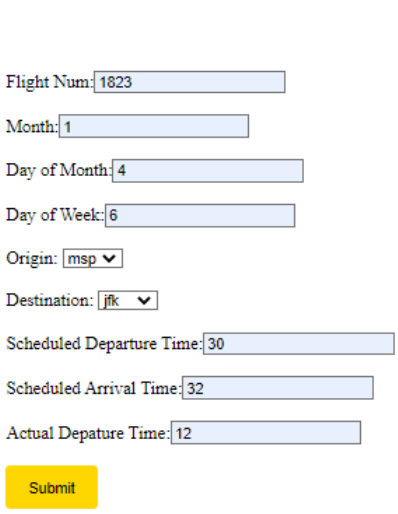
This is the part where we connect the model through flight.pkl in flask application and the index.html is where we have created the input forms for the website and that is also connected through the flask app.



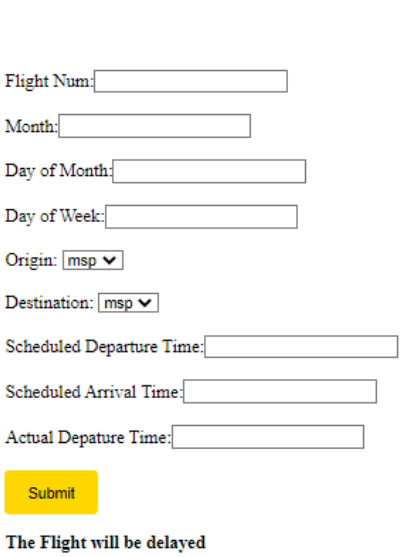

**We have given the inputs for predicting if the flight is getting delayed or not.**




**The output we get is "The Flight will be on time" so that means for the inputs that was given in the previous picture is that the flight won't get delayed apparently.**

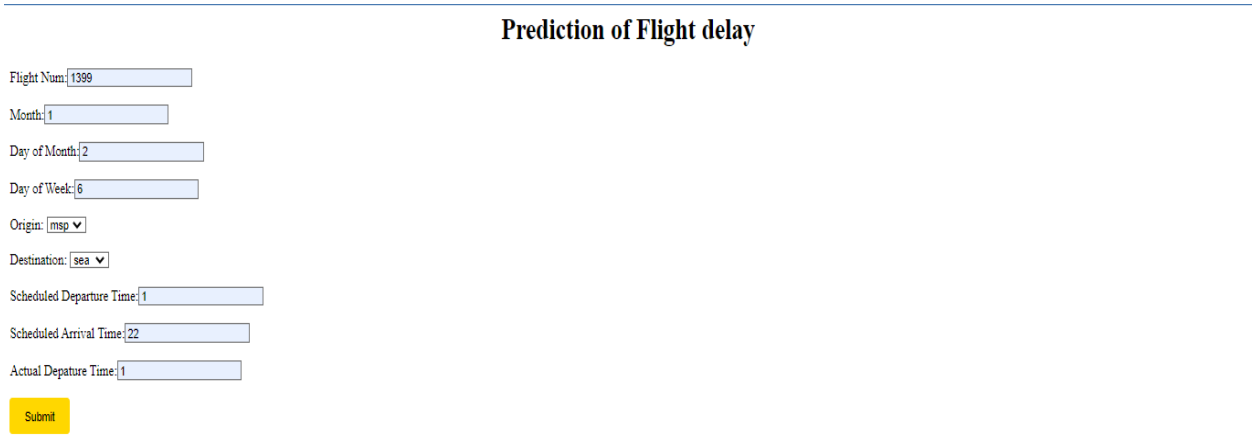



**We have given the inputs for predicting if the flight is getting delayed or not.**

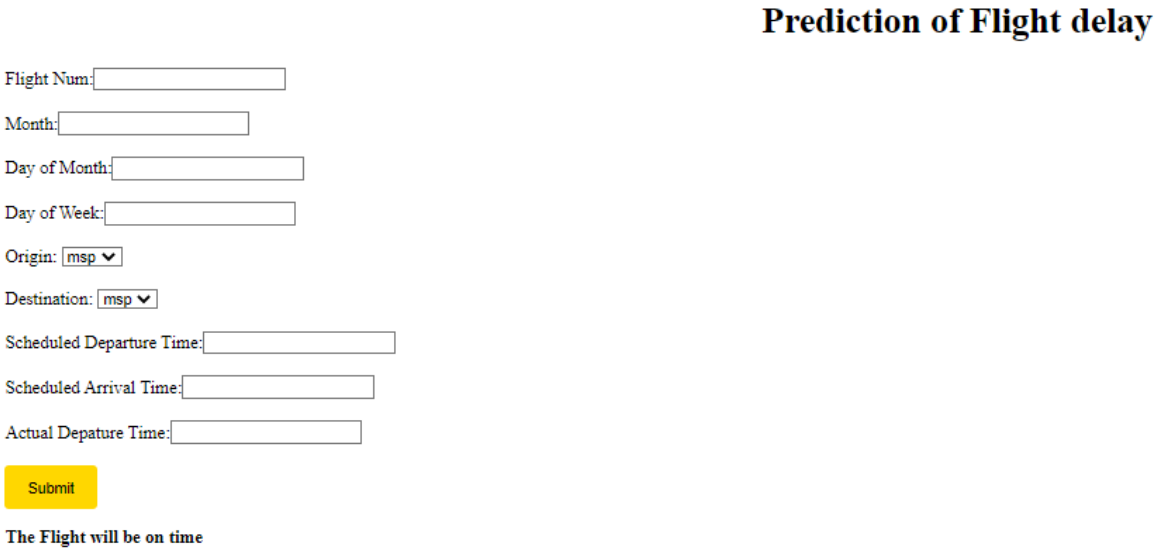



**The output we get is "The Flight will be delayed" so that means for the inputs that was given in the previous picture is that the flight will get delayed apparently.**

## 7. IBM DEPLOYMENT



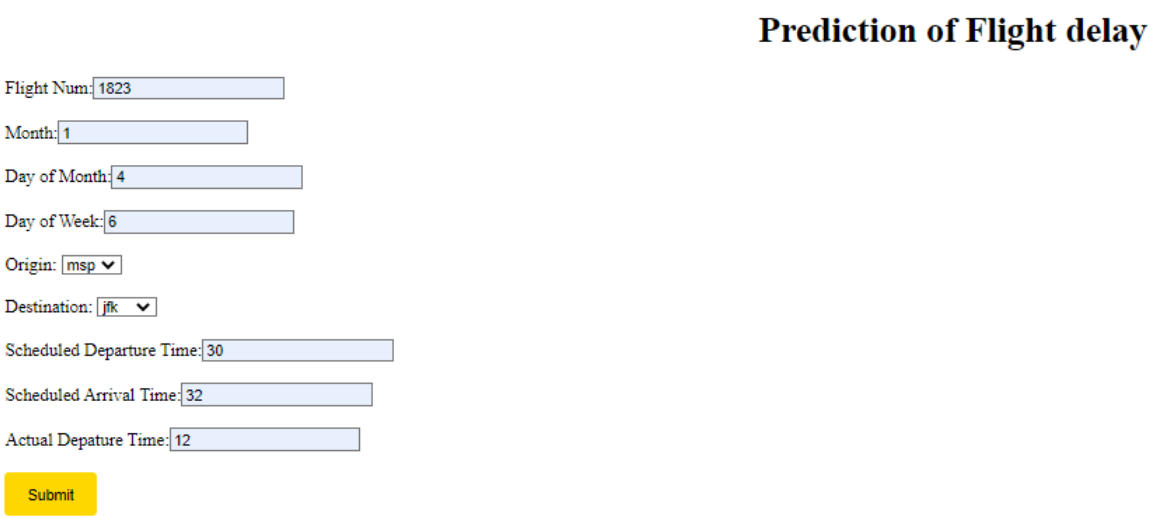

Next we are going to do this IBM Cloud Deployment, after deploying and connecting it to the flask application we are giving inputs to check if the flights are getting delayed or not.



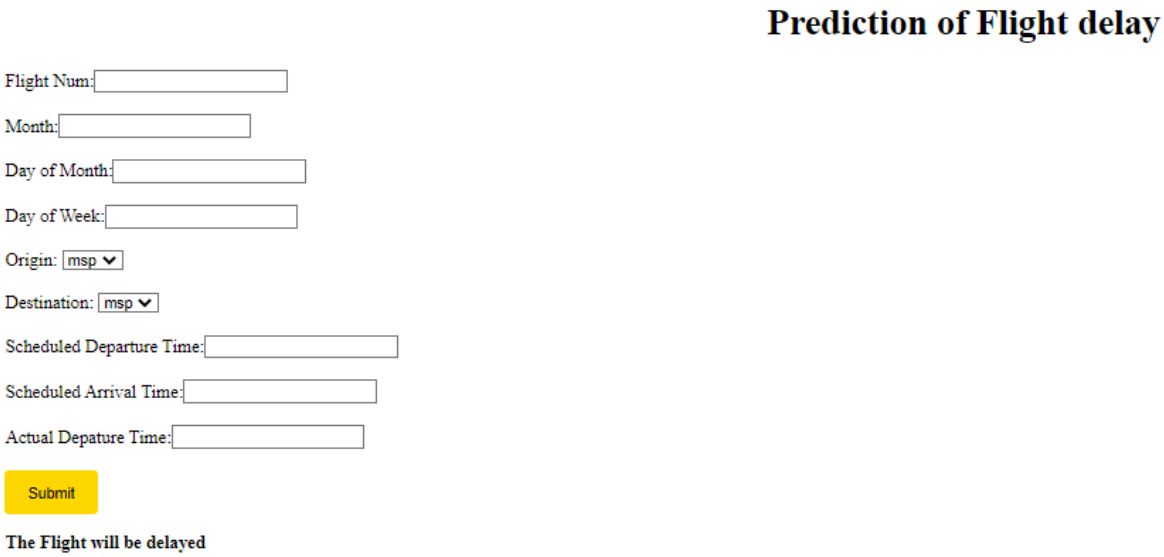

The output we get is “The Flight will be on time” so that means for the inputs that was given in the previous picture is that the flight won’t get delayed apparently.

```
Scoring response
0.0
The Flight will be on time
```

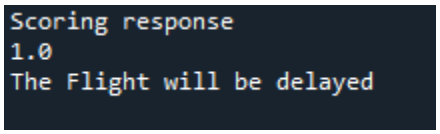
This output we get in Spyder application after we get the message from index.html if it's delayed or not.




We are giving inputs to check if the flights are getting delayed or not.



The output we get is “The Flight will be delayed” so that means for the inputs that was given in the previous picture is that the flight will get delayed apparently.




This output we get in Spyder application after we get the message from index.html saying the flight will be delayed.

## 8. Advantages

Using the flight delay system we can predict whether the flight will departure late when compared to the scheduled departure time.

## 9. Disadvantages

To use this system we need both scheduled departure time and actual departure time to calculate the delay.

## 10. Applications

This approach can benefit customers who have been waiting a long time for confirmation from customer service regarding the status of their flight's arrival or delay. By implementing this method, customers can receive prompt answers to their inquiries.

## 11. Conclusion

Predicting flight delays is an interesting research topic and required much attention these years. Majority of researchers have tried to develop and expand their models in order to increase the precision and accuracy of predicting flight delays. Since the issue of flights being on-time is very important, flight delay prediction models must have high precision and accuracy.
We have predicted the probability of flight delays by adapting it into a machine learning problem.

The prediction was made using a binary classification method of supervised machine learning. For the same, we used a decision tree model. The outcome of comparing algorithms reveals that ensemble algorithms based on trees typically perform better at forecasting flight delays for this data set. Repeating similar experimental procedures with more tree-based ensemble algorithms will be beneficial to understand their importance in predicting flight delays.

## 12. Future Work

The data analysis for this project dates back to 2008. The years 1987 to 2008 are covered by a sizable dataset, but managing a larger dataset necessitates extensive data pretreatment and cleaning. As a result, this project's future development will involve adding a bigger dataset. There are other options to preprocess a bigger dataset, such as deploying a Spark cluster on a computer or processing the data using cloud services like AWS and Azure. We can now employ neural networks algorithms on aviation and meteorological data thanks to recent developments in deep learning.

The pattern matching mechanism is used by neural networks. For data modeling, it is organized into three fundamental sections: feed-forward networks, feedback networks, and self-organizing networks. In comparison to self-organizing networks, which are typically employed in cluster analysis, feed-forward and feedback networks are typically utilized in prediction, pattern recognition, associative memory, and optimization calculations. Distributed computer

architecture with significant learning capabilities is provided by neural networks, which may express nonlinear interactions.

The project's focus is mostly on aircraft and meteorological data for the United States, but we can also incorporate data from other nations like China, India, and Russia. We may broaden the project's reach by including flight information from international flights rather than simply domestic ones.

## 13. Bibliography


1) Smartinternz Portal
2) How To Make a Web Application Using Flask in Python 3 (In Digital Ocean Website)
3) Decision Tree Model (In kdnuggets.com Website)
4) IBM CLOUD

Appendix:

Jupyter Notebook:

In [24]:
```
#x = dataset.iloc[:,0:16].values
y = dataset.iloc[:,5:6].values
#y
```

In [25]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

In [26]:
```
x_test.shape
```

Out[26]: (2247, 16)

In [27]:
```
x_train.shape
```

Out[27]: (8984, 16)

In [28]:
```
y_test.shape
```

Out[28]: (2247, 1)

In [29]:
```
y_train.shape
```

Out[29]: (8984, 1)

In [30]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [35]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(random_state = 0)
classifier.fit(x_train_smote,y_train_smote)
```

Out[35]:
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [36]:
```
decisiontree = classifier.predict(x_test)
```

In [37]:
```
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test,decisiontree)
acc
```

Out[37]: 1.0

In [38]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,decisiontree)
```

In [39]:
```
cm
```

Out[39]:
```
array([[1938,    0],
       [   0,  309]], dtype=int64)
```

In [40]:
```
import pickle
pickle.dump(classifier,open("flight.pkl","wb"))
```

**App.py code**

```
from flask import Flask,render_template,request
import numpy as np
import pandas as pd
import pickle
import os

model = pickle.load(open('flight.pkl','rb'))
app = Flask(__name__)

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



@app.route('/prediction',methods = ['POST'])

def predict():
  name = request.form['Flight Num']
  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 == "atl"):
    origin1,origin2,origin3,origin4,origin5 = 0,0,0,1,0
```

```
  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 == "atl"):
    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,destina
tion2,destination3,destination4,destination5,int(arrtime),int(dept15)]]
  y_pred = model.predict(total)

  print(y_pred)

  if(y_pred== [0.]):
    ans = "The Flight will be on time"
  else:
    ans = "The Flight will be delayed"
  return render_template("index.html",showcase = ans)

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

**Index.HTML Code**

```
<html>
<head>
<style>
h1 {text-align: center;}
p {text-align: center;}
div {text-align: center;}
input[type=submit]{
  background-color: #FFD700;
   padding: 12px 20px;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}
</style>

</head>

<body>
<h1>Prediction of Flight delay</h1>

<form action="/prediction" method="POST">
  <label for="fno">Name:<input type="text" id="fno" name="Flight Num"></label><br><br>
  <label for="mo">Month:<input type="text" id="mo" name="month"></label><br><br>
  <label for="Dmo">Day of Month:<input type="text" id="Dmo"
name="dayofmonth"></label><br><br>
  <label for="Dmw">Day of Week:<input type="text" id="Dmw"
name="dayofweek"></label><br><br>
  <label for="ori">Origin:</label>

<select name="origin" id="ori">
  <option value="msp">msp</option>
  <option value="dtw">dtw</option>
```

```
  <option value="jfk">jfk</option>
  <option value="sea">sea</option>
  <option value="atl">atl</option>
</select><br><br>

  <label for="Des">Destination:</label>

<select name="destination" id="Des">
  <option value="msp">msp</option>
  <option value="dtw">dtw</option>
  <option value="jfk">jfk</option>
  <option value="sea">sea</option>
  <option value="atl">atl</option>
</select><br><br>
<label for="SDT">Scheduled Departure Time:<input type="text" id="SDT"
name="dept"></label><br><br>
<label for="SAT">Scheduled Arrival Time:<input type="text" id="SAT"
name="arrtime"></label><br><br>
<label for="AAT">Actual Depature Time:<input type="text" id="AAT"
name="actdept"></label><br><br>
  <input type="submit" value="Submit">
</form>
<b>{{showcase}}</b>

</body>
</html>
```

**App_ibm.py Code**

```
from flask import Flask,render_template,request

import pickle


import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM
Cloud account.
API_KEY = "sB-WmbroZW3OEXJtEdp_uajA4ka1azanV1ZvLEXgLpQN"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}



model = pickle.load(open('flight.pkl','rb'))
app = Flask(__name__)

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


@app.route('/prediction',methods = ['POST'])

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 == "atl"):
   origin1,origin2,origin3,origin4,origin5 = 0,0,0,1,0

 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 == "atl"):
   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,destina
tion2,destination3,destination4,destination5,int(arrtime),int(dept15)]]
 # y_pred = model.predict(total)
  # NOTE: manually define and pass the array(s) of values to be scored in the next line
```

```
  payload_scoring = {"input_data": [{"fields":
["name","month","dayofmonth","dayofweek","origin1","origin2","origin3","origin4","origin5","d
estination1","destination2","destination3","destination4","destination5","scheduleddeparturetime"
,"actualdeparturetime"],"values":total}]}

#payload_scoring = {"input_data": [{"fields": [array_of_input_fields], "values":
[array_of_values_to_be_scored, another_array_of_values_to_be_scored]}]}

  response_scoring =
requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/2fc12f58-ef31-48b0-a996-b
6a4b33ae6a2/predictions?version=2022-06-02', json=payload_scoring,
   headers={'Authorization': 'Bearer ' + mltoken})
  print("Scoring response")
  pred = response_scoring.json()
  #print(response_scoring.json())
  output=pred['predictions'][0]['values'][0][0]
  print(output)

 # print(y_pred)

  if(output==0):
     ans ="The Flight will be on time"
  else:
     ans ="The Flight will be delayed"
  print(ans)
  return render_template("index.html",showcase= ans)

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