Flight Delay Prediction using IBM Cloud

Done By :

Aswathy Ashok
Kristu Jyothi College of Management and Technology

1. Introduction

1.1 Overview :

In the present day scenario, Time is money. Flight delays end up hurting airports, passengers and airlines. Being able to predict how much delay a flight incurs will save passengers their precious time as well as hardships caused due to flight delays or in worse cases cancellations.

1.2 Purpose :

The main objective of the model is 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.

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 analysing 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.

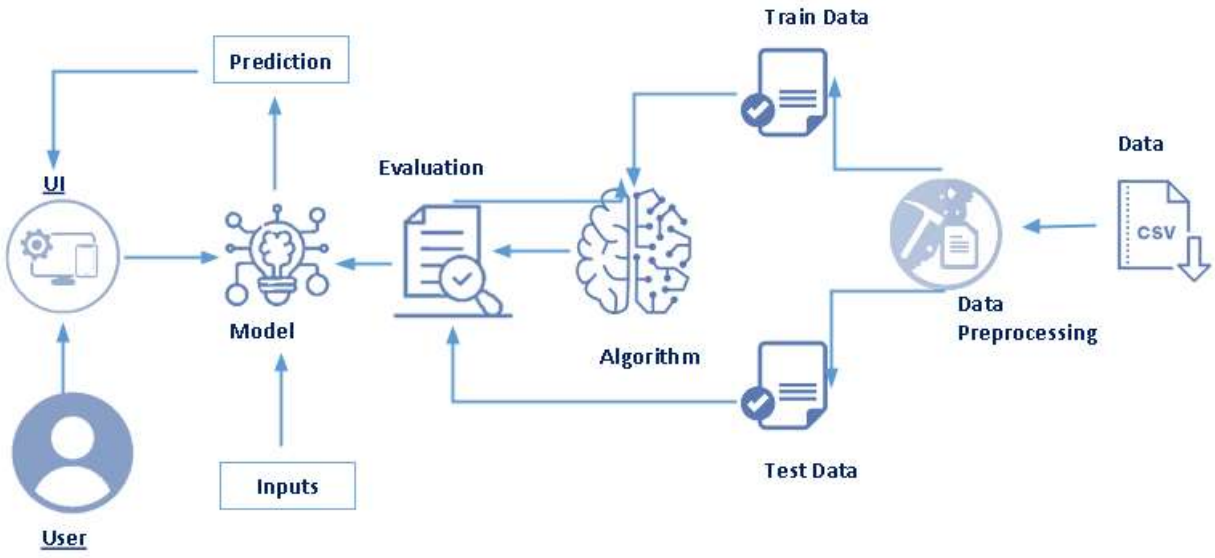
2.2 Proposed Solution

The input to our algorithm is rows of feature vectors like departure date, departure delay, distance between the two airports, scheduled arrival time etc. We then use a decision tree classifier to predict if the flight arrival will be delayed or not. A flight is considered to be delayed when the difference between scheduled and actual arrival times is greater

than 15 minutes. Furthermore, we compare decision tree classifiers with logistic regression and a simple neural network for various figures of merit. Then we deploy it with flask through IBM Cloud

3. Theoretical Analytics

3.1 Block Diagram



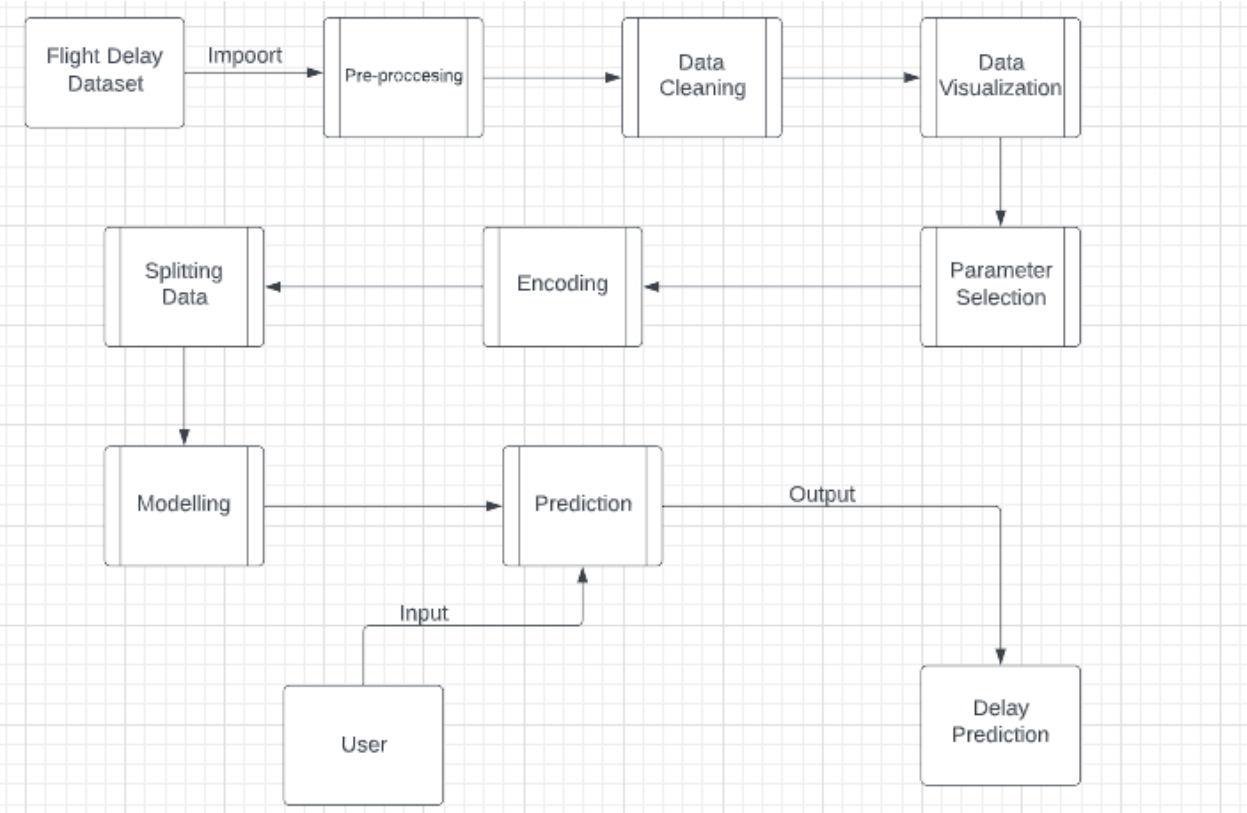
3.2. Hardware and Software

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

4. Analysis

While working on the model we get to find out the calculations of flight delays are being carried out. Plus we get to know how a particular machine learning model will help finding out the delay process of a flight. We get to know how mainstream airline websites work and how people check if their flight is delayed or not.

5. Flowchart



6. Result

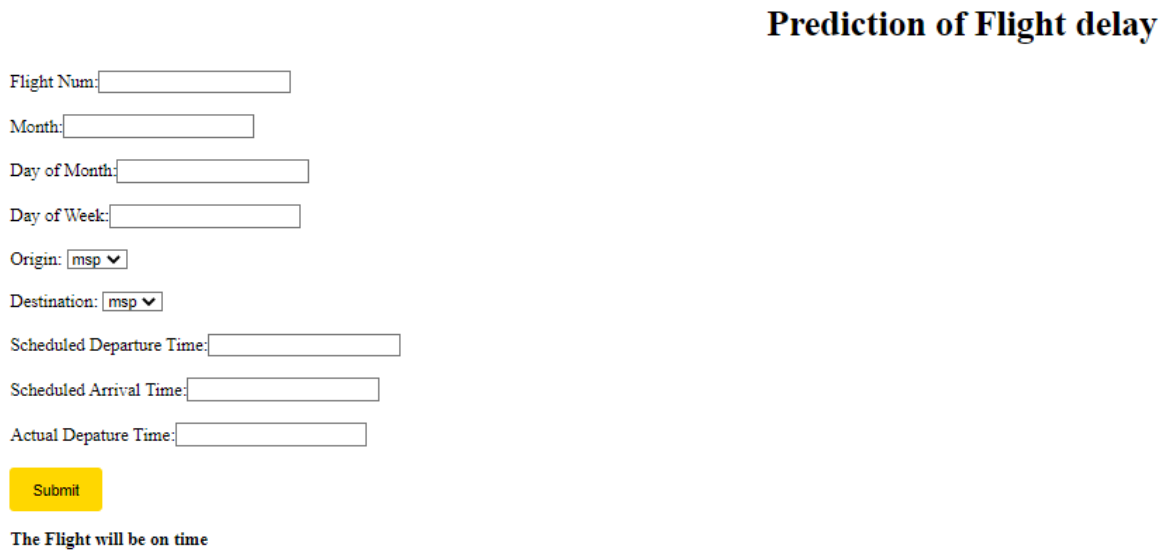
```
In [35]: from sklearn.tree import DecisionTreeClassifier
         classifier = DecisionTreeClassifier(random_state = 0)
         classifier.fit(x_train_smote,y_train_smote)

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.

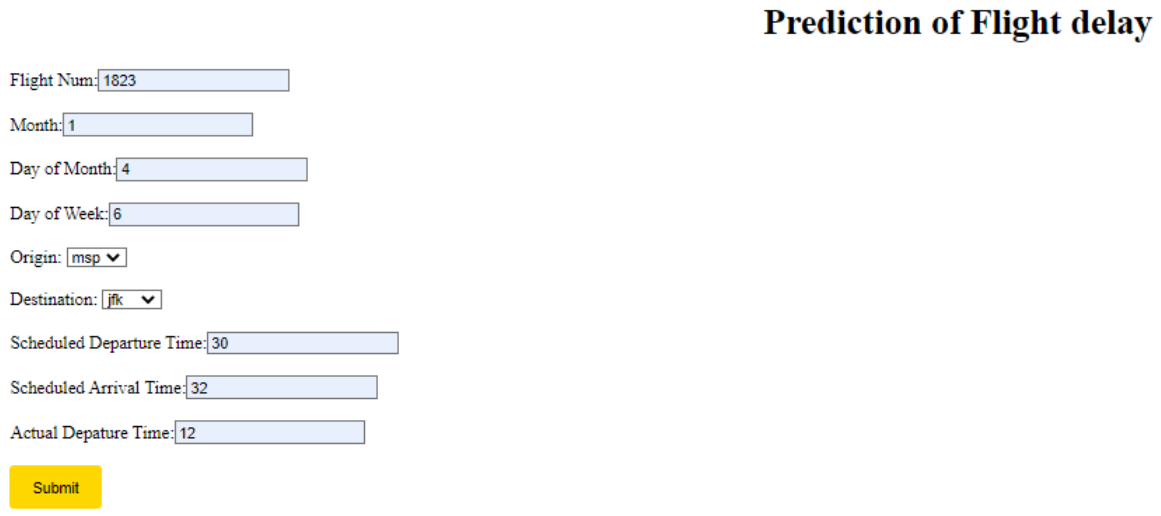
```
from flask import Flask,render_template,request
import numpy as np
import pandas as pd
import pickle
import os

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.

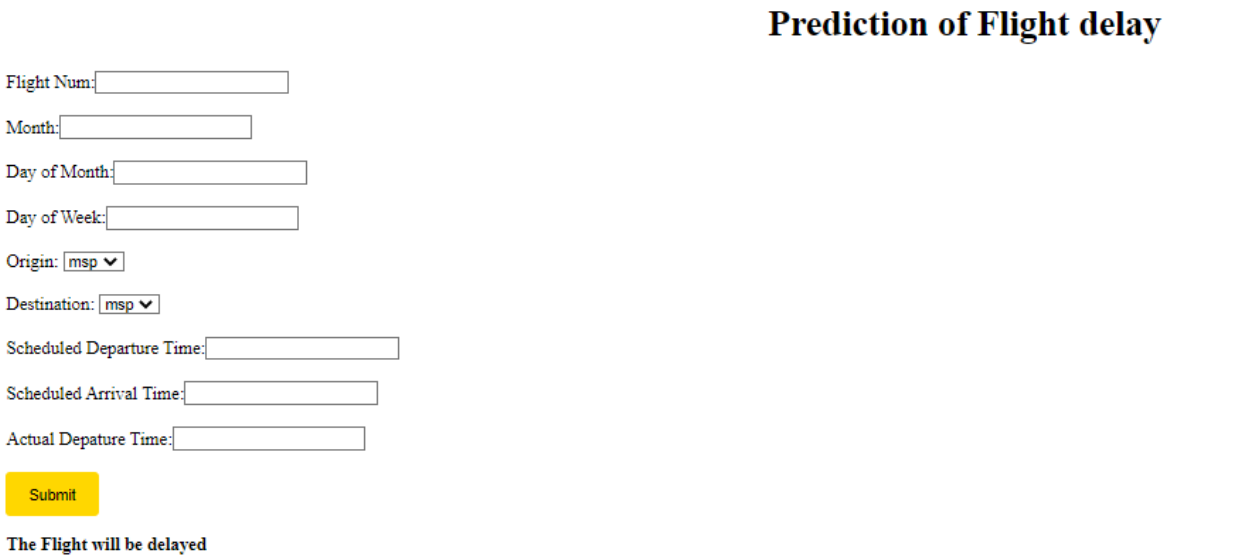
We have given the inputs for predicting if the flight is getting delayed or not.



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.



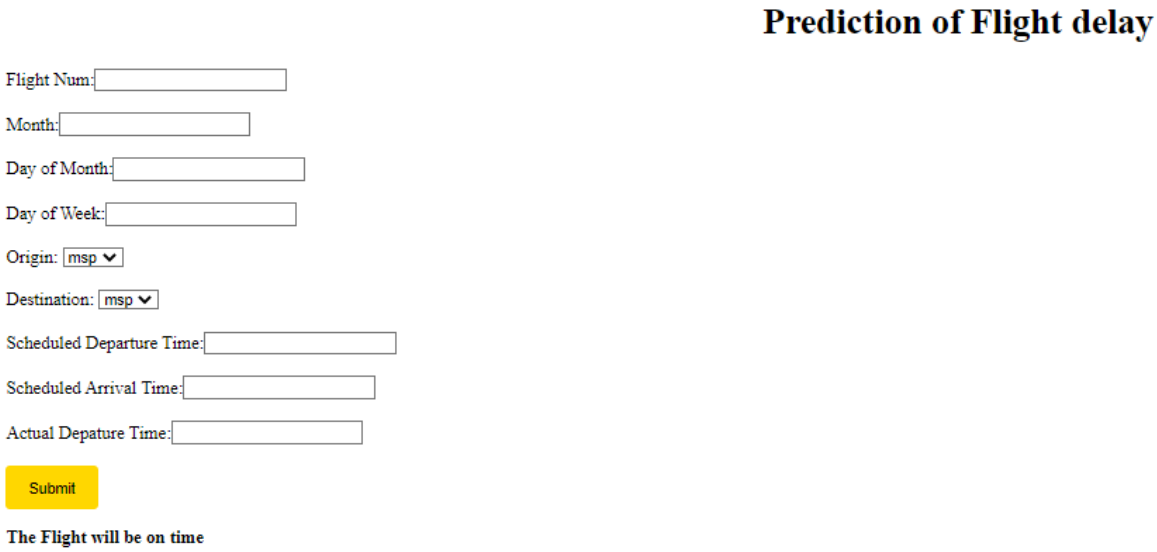
We have given the inputs for predicting if the flight is getting delayed or not.



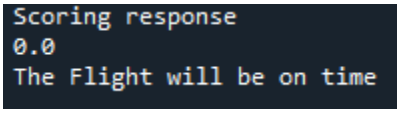
The output we get is "The Flight will be delayed" so that means for the inputs that was given in the previous picture is that the flight will get delayed apparently.

IBM DEPLOYMENT

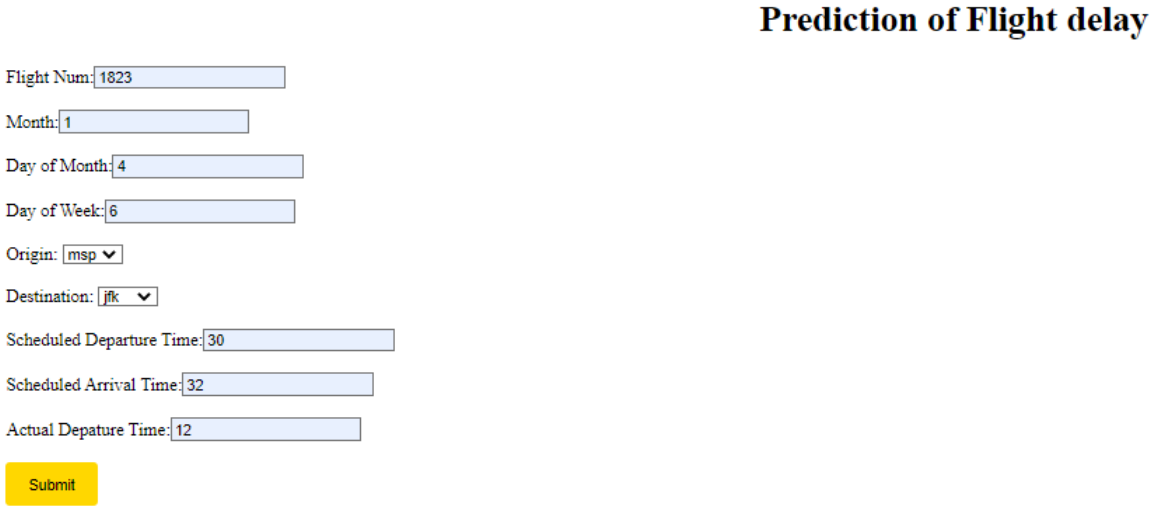
Next we are going to do this IBM Cloud Deployment, after deploying and connecting it to the flask application we are giving inputs to check if the flights are getting delayed or not.



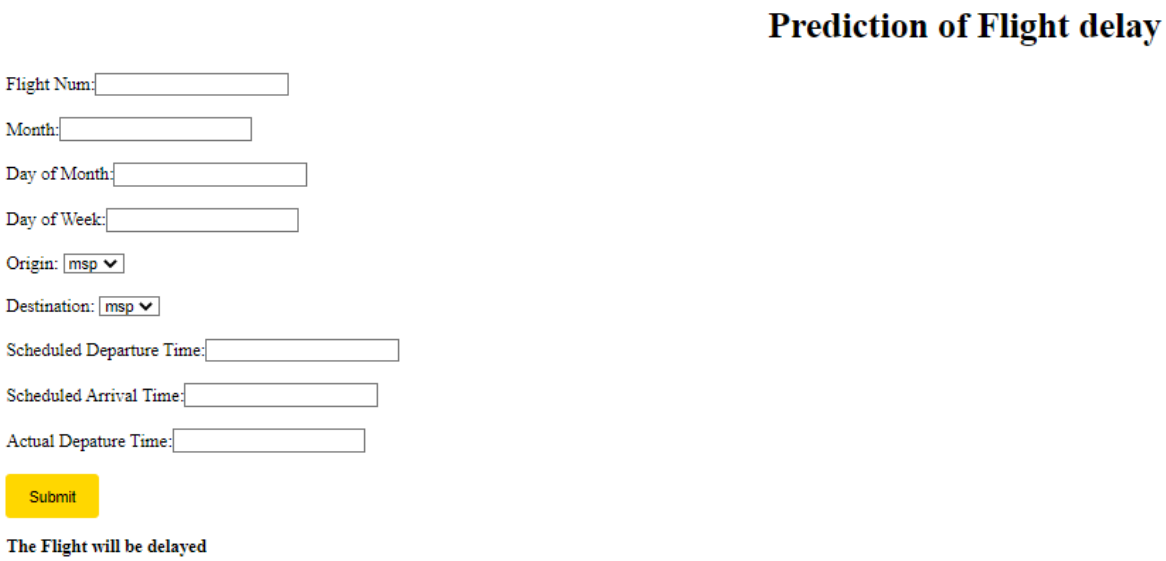
The output we get is "The Flight will be on time" so that means for the inputs that was given in the previous picture is that the flight won't get delayed apparently.



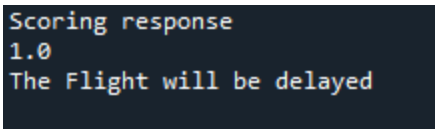
This output we get in Spyder application after we get the message from index.html if it's delayed or not.



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.

7. **Advantages**

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

   Disadvantages

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

8. **Applications**
   This can be applied for customers who wait for confirmation if the flight will arrive or will get delayed through customer service for a long time. Customers will get to know their answer pretty quick also.

9. **Conclusion**

Following this project, it is likely that the choice of approaches that can be utilised to produce notable results will be heavily influenced by the dataset's balance. Many machine learning models, such as Decision Tree Classifier, have been used to predict airplane arrival and delays. We were able to acquire a quick answer about the flight status thanks to IBM Cloud and the Flask application.

10. Future Work

Many machine learning models can be used to forecast airline arrival delays, including Logistic Regression, Random Forest Regression, Linear Regression, and its variation Boosted Linear Regression. Even these algorithms will be able to forecast delays with excellent accuracy when given the proper combination of input parameters. We can forecast arrival delay even without including departure delay as an attribute if weather and air traffic control information are made available. We can also estimate whether a flight will be delayed or cancelled depending on weather elements such as snow, rain, or storms.

11. 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,
destination2,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 requests

# NOTE: you must manually set API_KEY below using information retrieved from your
IBM Cloud account.
API_KEY = "UoONOA3v5bJkranlPkbW2jyrO8IhUpzafOviD1hc7AQg"
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,
destination2,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"
,"destination1","destination2","destination3","destination4","destination5","scheduleddep
arturetime","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-
b6a4b33ae6a2/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)
```