PROJECT DEVELOPMENT PHASE

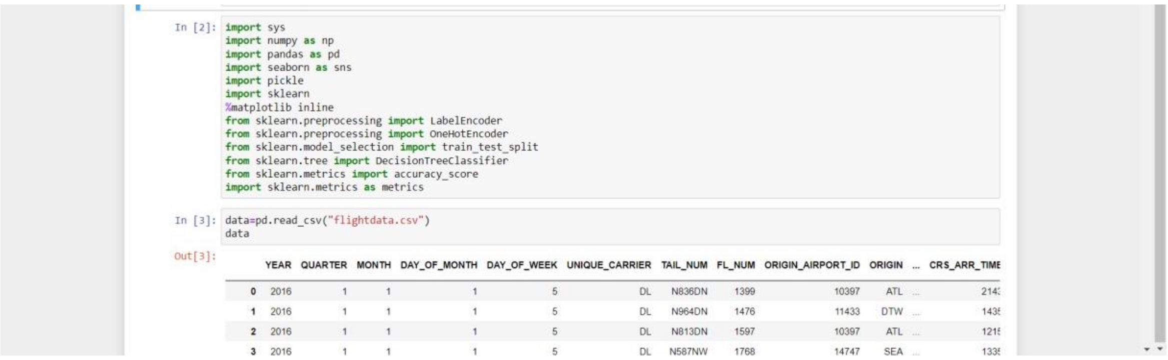
SPRINT 3 – CODE AND TESTCASE

Date	10 November 2022
Team ID	PNT2022TMID02840
Project	Flight delay prediction using Machine learning
Marks	8 Marks

In this Sprint development phase, we have create an model with the help of Pre-processed dataset. We have used Decision Tree Classifier Algorithm for model development. Also we have implement method to check the accuracy of our model and convert the model into pkl file by importing Pickle python library. With the help of pickle model file the prediction is performed by Flask App.

Jupyter notebook :

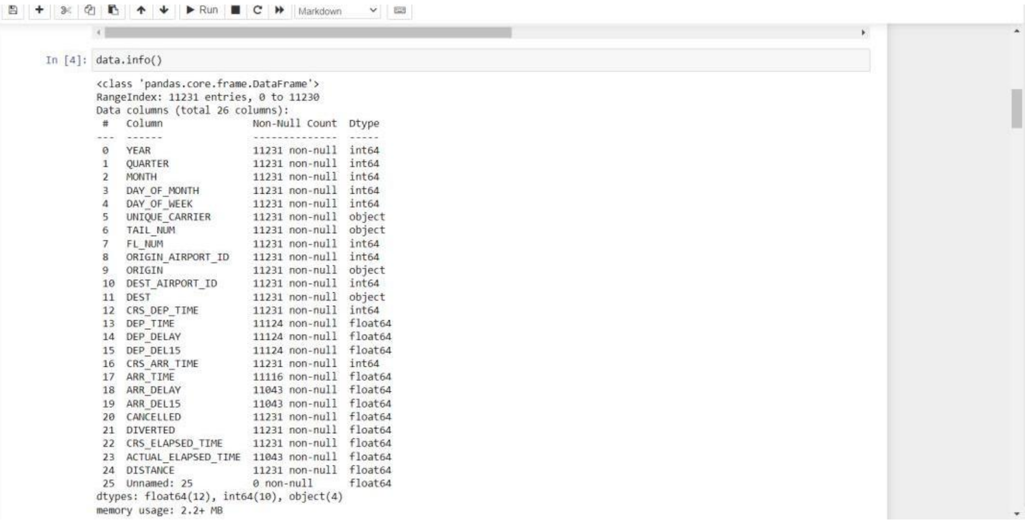
Screenshots :



```
In [2]: import sys
        import numpy as np
        import pandas as pd
        import seaborn as sns
        import pickle
        import sklearn
        %matplotlib inline
        from sklearn.preprocessing import LabelEncoder
        from sklearn.preprocessing import OneHotEncoder
        from sklearn.model_selection import train_test_split
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.metrics import accuracy_score
        import sklearn.metrics as metrics
```

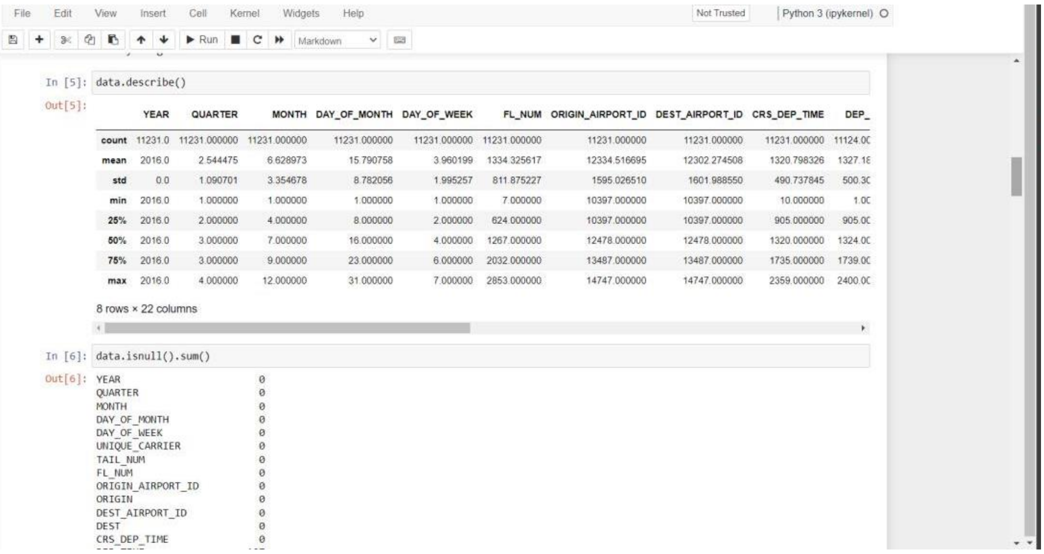
```
In [3]: data=pd.read_csv("flightdata.csv")
        data
```

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



In [4]:
```
data.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
```



In [5]:
```
data.describe()
```

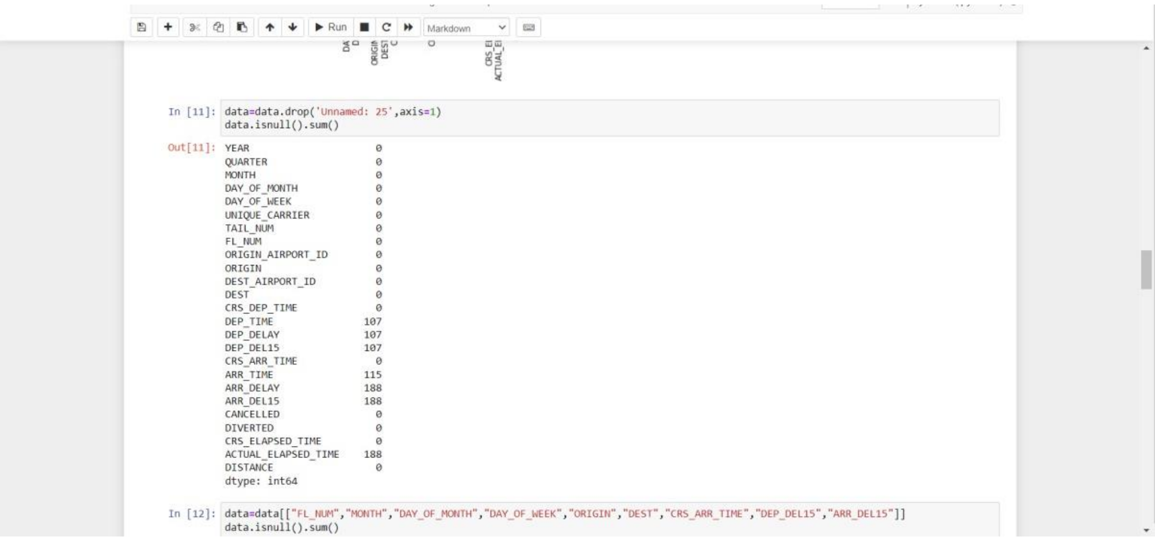
Out[5]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_NUM	ORIGIN_AIRPORT_ID	DEST_AIRPORT_ID	CRS_DEP_TIME	DEP_
count	11231.0	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11124.0
mean	2016.0	2.544475	6.628973	15.790758	3.960199	1334.325617	12334.516695	12302.274508	1320.798326	1327.1
std	0.0	1.090701	3.354678	8.782056	1.995257	811.875227	1595.026510	1601.988550	490.737845	500.3
min	2016.0	1.000000	1.000000	1.000000	1.000000	7.000000	10397.000000	10397.000000	10.000000	1.0
25%	2016.0	2.000000	4.000000	8.000000	2.000000	624.000000	10397.000000	10397.000000	905.000000	905.0
50%	2016.0	3.000000	7.000000	16.000000	4.000000	1267.000000	12478.000000	12478.000000	1320.000000	1324.0
75%	2016.0	3.000000	9.000000	23.000000	6.000000	2032.000000	13487.000000	13487.000000	1735.000000	1739.0
max	2016.0	4.000000	12.000000	31.000000	7.000000	2853.000000	14747.000000	14747.000000	2359.000000	2400.0

8 rows × 22 columns

In [6]:
```
data.isnull().sum()
```

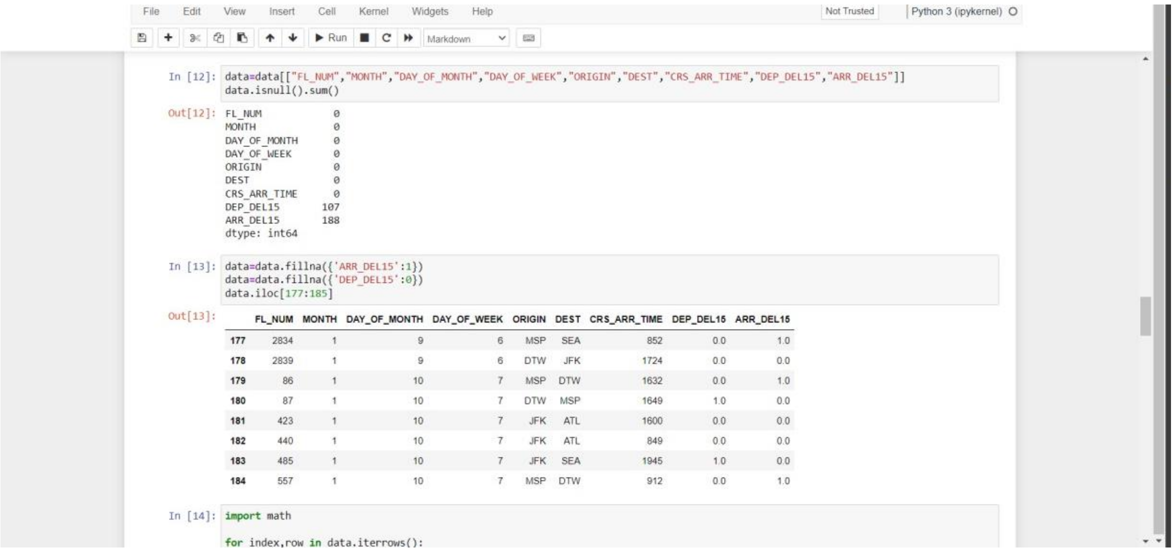
Out[6]:
```
YEAR                     0
QUARTER                  0
MONTH                    0
DAY_OF_MONTH             0
DAY_OF_WEEK              0
UNIQUE_CARRIER           0
TAIL_NUM                 0
FL_NUM                   0
ORIGIN_AIRPORT_ID        0
ORIGIN                   0
DEST_AIRPORT_ID          0
DEST                     0
CRS_DEP_TIME             0
```



```
In [11]: data=data.drop('Unnamed: 25',axis=1)
         data.isnull().sum()
```

```
Out[11]: YEAR                     0
         QUARTER                  0
         MONTH                    0
         DAY_OF_MONTH             0
         DAY_OF_WEEK              0
         UNIQUE_CARRIER           0
         TAIL_NUM                 0
         FL_NUM                   0
         ORIGIN_AIRPORT_ID        0
         ORIGIN                   0
         DEST_AIRPORT_ID          0
         DEST                     0
         CRS_DEP_TIME             0
         DEP_TIME               107
         DEP_DELAY              107
         DEP_DEL15              107
         CRS_ARR_TIME             0
         ARR_TIME               115
         ARR_DELAY              188
         ARR_DEL15              188
         CANCELLED                0
         DIVERTED                 0
         CRS_ELAPSED_TIME         0
         ACTUAL_ELAPSED_TIME    188
         DISTANCE                 0
         dtype: int64
```

```
In [12]: data=data[["FL_NUM","MONTH","DAY_OF_MONTH","DAY_OF_WEEK","ORIGIN","DEST","CRS_ARR_TIME","DEP_DEL15","ARR_DEL15"]]
         data.isnull().sum()
```



In [12]:

```
data=data[["FL_NUM","MONTH","DAY_OF_MONTH","DAY_OF_WEEK","ORIGIN","DEST","CRS_ARR_TIME","DEP_DEL15","ARR_DEL15"]]
data.isnull().sum()
```

Out[12]:

```
FL_NUM            0
MONTH             0
DAY_OF_MONTH      0
DAY_OF_WEEK       0
ORIGIN            0
DEST              0
CRS_ARR_TIME      0
DEP_DEL15       107
ARR_DEL15       188
dtype: int64
```

In [13]:

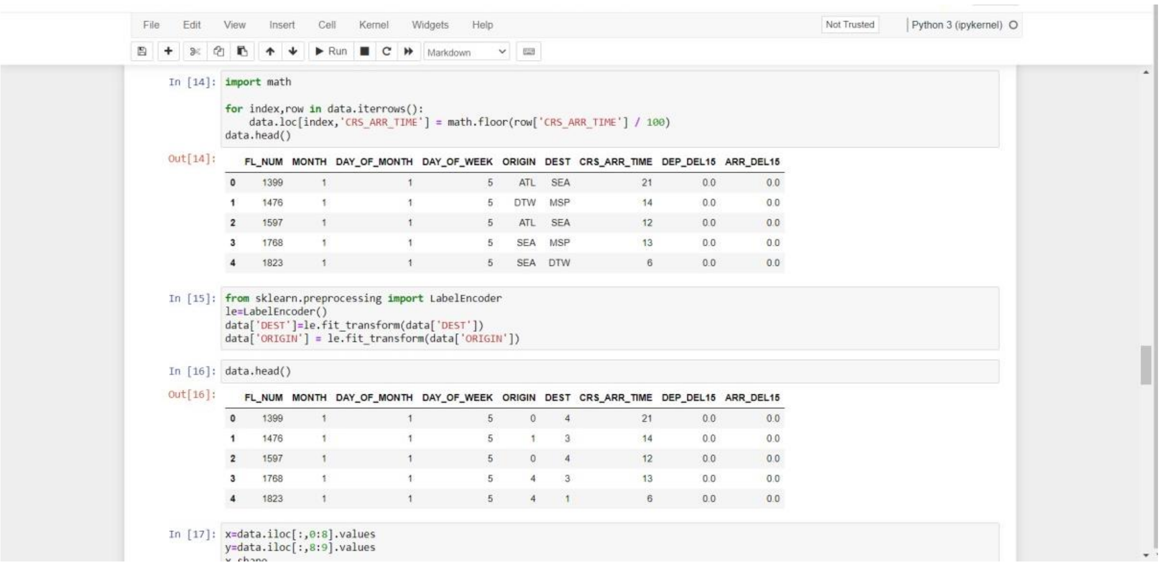
```
data=data.fillna({'ARR_DEL15':1})
data=data.fillna({'DEP_DEL15':0})
data.iloc[177:185]
```

Out[13]:

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
177	2834	1	9	6	MSP	SEA	852	0.0	1.0
178	2839	1	9	6	DTW	JFK	1724	0.0	0.0
179	86	1	10	7	MSP	DTW	1632	0.0	1.0
180	87	1	10	7	DTW	MSP	1649	1.0	0.0
181	423	1	10	7	JFK	ATL	1600	0.0	0.0
182	440	1	10	7	JFK	ATL	849	0.0	0.0
183	485	1	10	7	JFK	SEA	1945	1.0	0.0
184	557	1	10	7	MSP	DTW	912	0.0	1.0

In [14]:

```
import math

for index,row in data.iterrows():
```



```
In [14]: import math

         for index,row in data.iterrows():
             data.loc[index,'CRS_ARR_TIME'] = math.floor(row['CRS_ARR_TIME'] / 100)
         data.head()
```

Out[14]:

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
0	1399	1	1	5	ATL	SEA	21	0.0	0.0
1	1476	1	1	5	DTW	MSP	14	0.0	0.0
2	1597	1	1	5	ATL	SEA	12	0.0	0.0
3	1768	1	1	5	SEA	MSP	13	0.0	0.0
4	1823	1	1	5	SEA	DTW	6	0.0	0.0

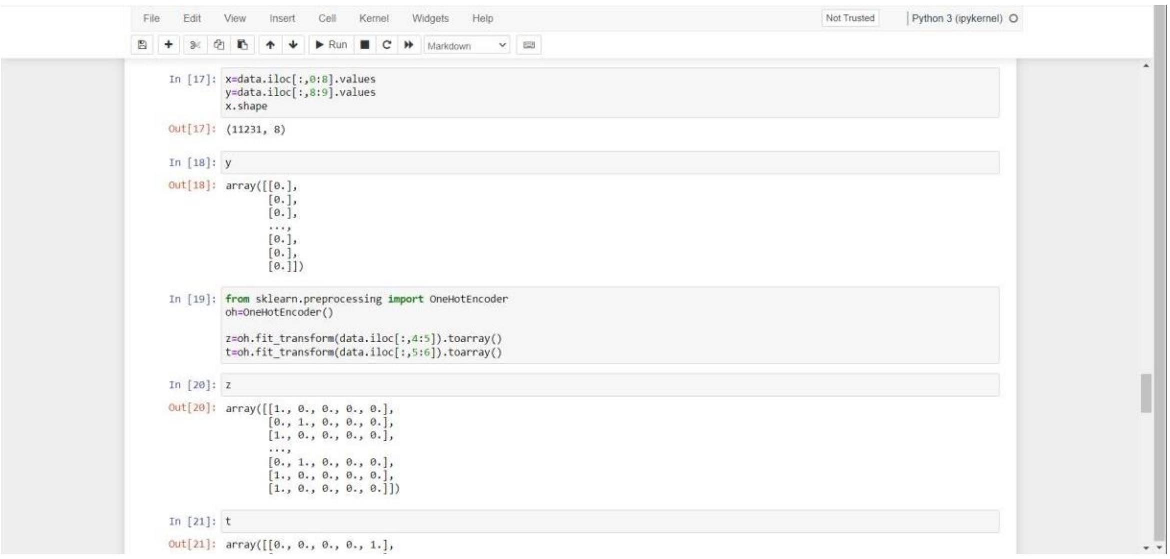
```
In [15]: from sklearn.preprocessing import LabelEncoder
         le=LabelEncoder()
         data['DEST']=le.fit_transform(data['DEST'])
         data['ORIGIN'] = le.fit_transform(data['ORIGIN'])
```

```
In [16]: data.head()
```

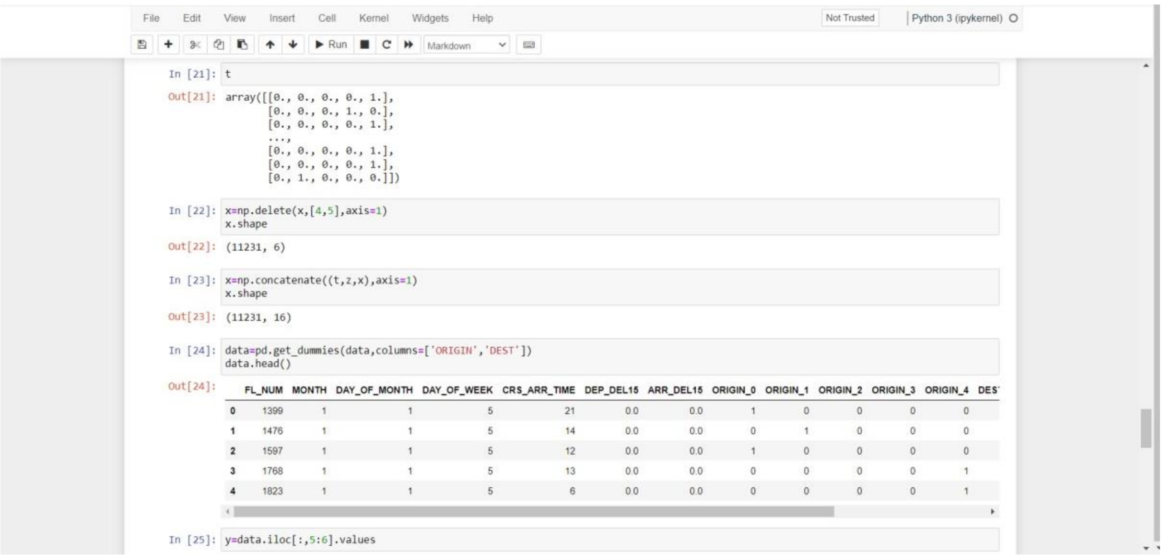
Out[16]:

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
0	1399	1	1	5	0	4	21	0.0	0.0
1	1476	1	1	5	1	3	14	0.0	0.0
2	1597	1	1	5	0	4	12	0.0	0.0
3	1768	1	1	5	4	3	13	0.0	0.0
4	1823	1	1	5	4	1	6	0.0	0.0

```
In [17]: x=data.iloc[:,0:8].values
         y=data.iloc[:,8:9].values
         x.shape
```



```
In [17]: x=data.iloc[:,0:8].values
         y=data.iloc[:,8:9].values
         x.shape

Out[17]: (11231, 8)

In [18]: y

Out[18]: array([[0.],
                [0.],
                [0.],
                ...,
                [0.],
                [0.],
                [0.]])

In [19]: from sklearn.preprocessing import OneHotEncoder
         oh=OneHotEncoder()

         z=oh.fit_transform(data.iloc[:,4:5]).toarray()
         t=oh.fit_transform(data.iloc[:,5:6]).toarray()

In [20]: z

Out[20]: array([[1., 0., 0., 0., 0.],
                [0., 1., 0., 0., 0.],
                [1., 0., 0., 0., 0.],
                ...,
                [0., 1., 0., 0., 0.],
                [1., 0., 0., 0., 0.],
                [1., 0., 0., 0., 0.]])

In [21]: t

Out[21]: array([[0., 0., 0., 0., 1.],
```



```
In [21]: t
```

```
Out[21]: array([[0., 0., 0., 0., 1.],
                [0., 0., 0., 1., 0.],
                [0., 0., 0., 0., 1.],
                ...,
                [0., 0., 0., 0., 1.],
                [0., 0., 0., 0., 1.],
                [0., 1., 0., 0., 0.]])
```

```
In [22]: x=np.delete(x,[4,5],axis=1)
         x.shape
```

```
Out[22]: (11231, 6)
```

```
In [23]: x=np.concatenate((t,z,x),axis=1)
         x.shape
```

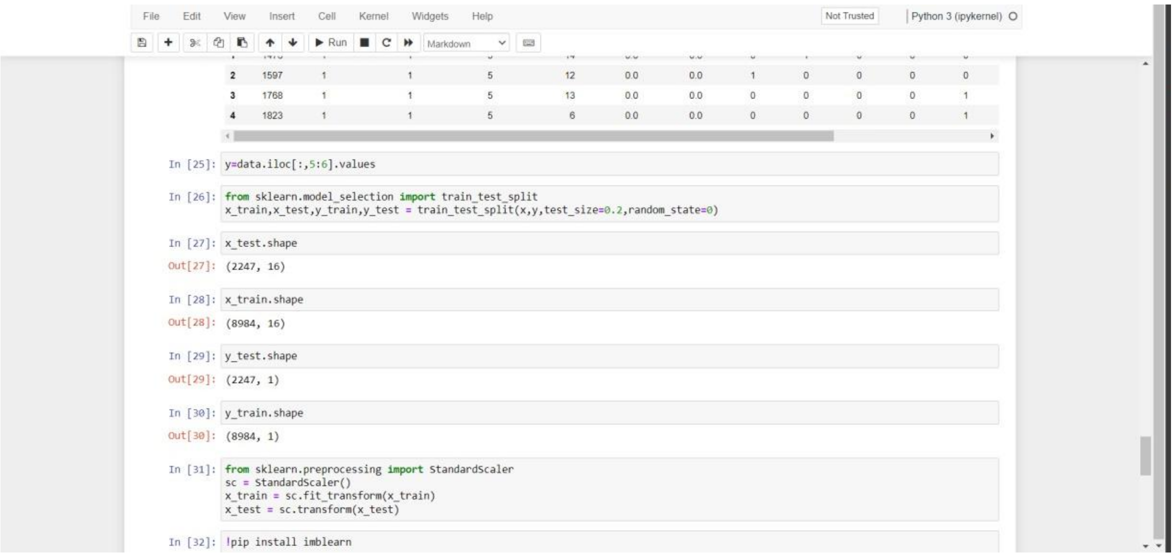
```
Out[23]: (11231, 16)
```

```
In [24]: data=pd.get_dummies(data,columns=['ORIGIN','DEST'])
         data.head()
```

Out[24]:

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15	ORIGIN_0	ORIGIN_1	ORIGIN_2	ORIGIN_3	ORIGIN_4	DES
0	1399	1	1	5	21	0.0	0.0	1	0	0	0	0	
1	1476	1	1	5	14	0.0	0.0	0	1	0	0	0	
2	1597	1	1	5	12	0.0	0.0	1	0	0	0	0	
3	1768	1	1	5	13	0.0	0.0	0	0	0	0	1	
4	1823	1	1	5	6	0.0	0.0	0	0	0	0	1	

```
In [25]: y=data.iloc[:,5:6].values
```



2	1597	1	1	5	12	0.0	0.0	1	0	0	0	0
3	1768	1	1	5	13	0.0	0.0	0	0	0	0	1
4	1823	1	1	5	6	0.0	0.0	0	0	0	0	1

```
In [25]: y=data.iloc[:,5:6].values
```

```
In [26]: 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 [27]: x_test.shape
```

```
Out[27]: (2247, 16)
```

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

```
Out[28]: (8984, 16)
```

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

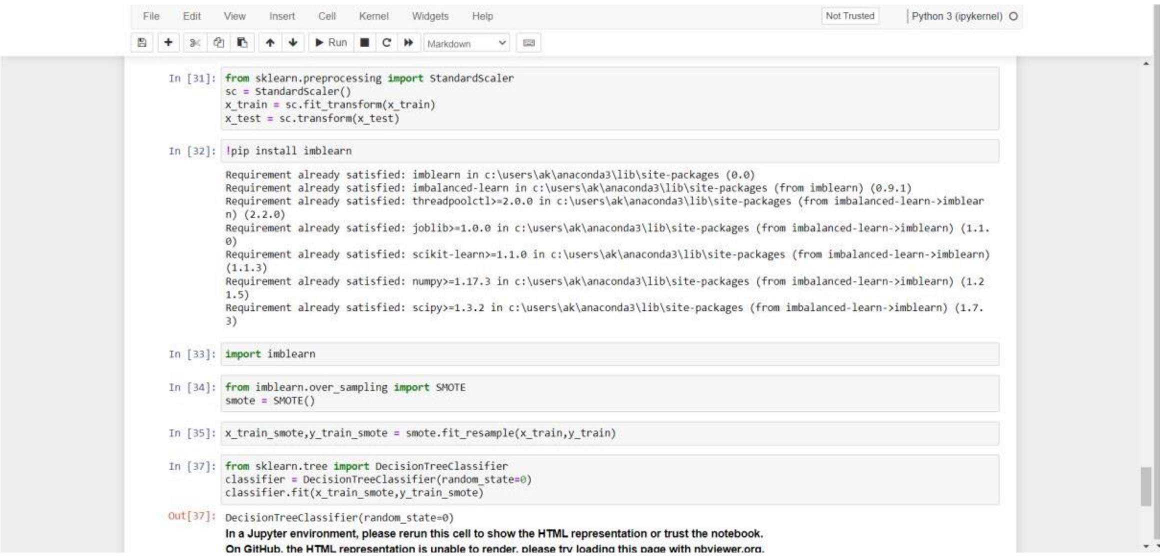
```
Out[29]: (2247, 1)
```

```
In [30]: y_train.shape
```

```
Out[30]: (8984, 1)
```

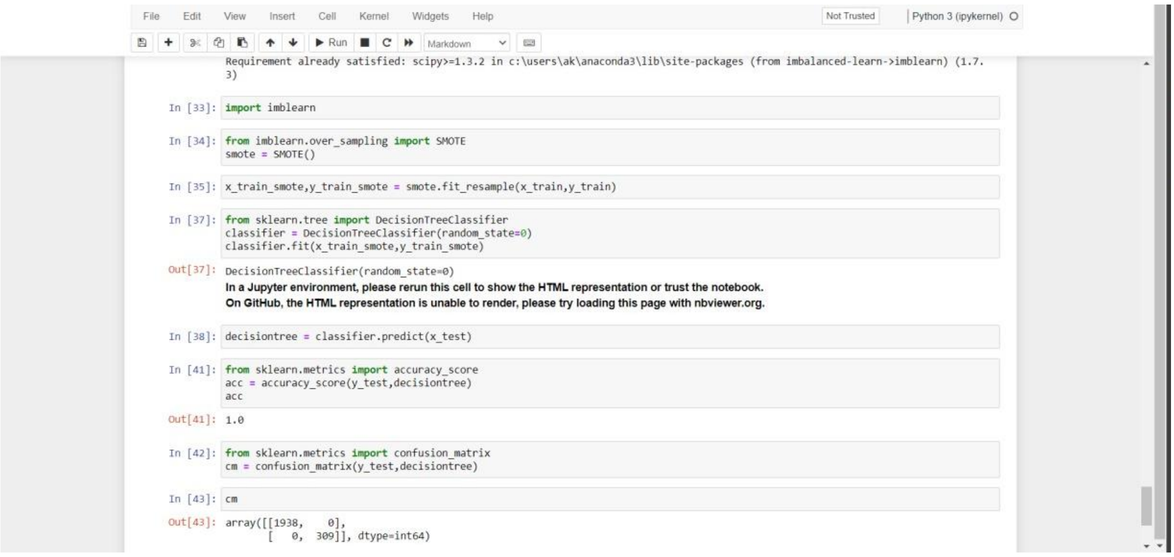
```
In [31]: from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         x_train = sc.fit_transform(x_train)
         x_test = sc.transform(x_test)
```

```
In [32]: !pip install imblearn
```



```
In [31]: from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         x_train = sc.fit_transform(x_train)
         x_test = sc.transform(x_test)

In [32]: !pip install imblearn

         Requirement already satisfied: imblearn in c:\users\ak\anaconda3\lib\site-packages (0.0)
         Requirement already satisfied: imbalanced-learn in c:\users\ak\anaconda3\lib\site-packages (from imblearn) (0.9.1)
         Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\ak\anaconda3\lib\site-packages (from imbalanced-learn->imblear
         n) (2.2.0)
         Requirement already satisfied: joblib>=1.0.0 in c:\users\ak\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.1.
         0)
         Requirement already satisfied: scikit-learn>=1.1.0 in c:\users\ak\anaconda3\lib\site-packages (from imbalanced-learn->imblearn)
         (1.1.3)
         Requirement already satisfied: numpy>=1.17.3 in c:\users\ak\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.2
         1.5)
         Requirement already satisfied: scipy>=1.3.2 in c:\users\ak\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.7.
         3)

In [33]: import imblearn

In [34]: from imblearn.over_sampling import SMOTE
         smote = SMOTE()

In [35]: x_train_smote,y_train_smote = smote.fit_resample(x_train,y_train)

In [37]: from sklearn.tree import DecisionTreeClassifier
         classifier = DecisionTreeClassifier(random_state=0)
         classifier.fit(x_train_smote,y_train_smote)

Out[37]: DecisionTreeClassifier(random_state=0)
```

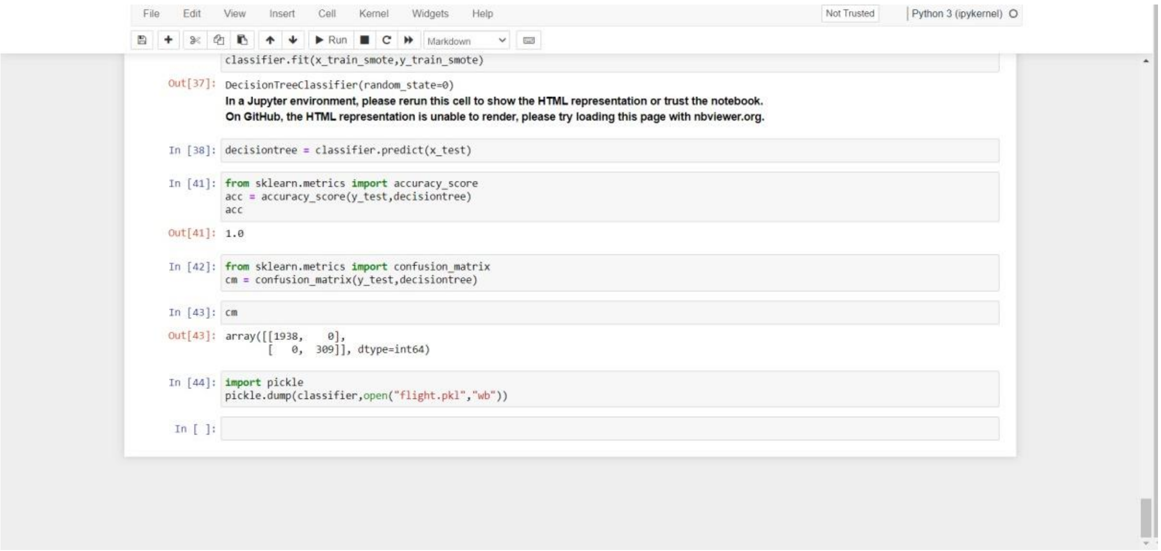
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.



```
         Requirement already satisfied: scipy>=1.3.2 in c:\users\ak\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.7.
         3)

In [33]: import imblearn

In [34]: from imblearn.over_sampling import SMOTE
         smote = SMOTE()

In [35]: x_train_smote,y_train_smote = smote.fit_resample(x_train,y_train)

In [37]: from sklearn.tree import DecisionTreeClassifier
         classifier = DecisionTreeClassifier(random_state=0)
         classifier.fit(x_train_smote,y_train_smote)

Out[37]: DecisionTreeClassifier(random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [38]: decisiontree = classifier.predict(x_test)

In [41]: from sklearn.metrics import accuracy_score
         acc = accuracy_score(y_test,decisiontree)
         acc

Out[41]: 1.0

In [42]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test,decisiontree)

In [43]: cm

Out[43]: array([[1938,    0],
                [   0,  309]], dtype=int64)
```

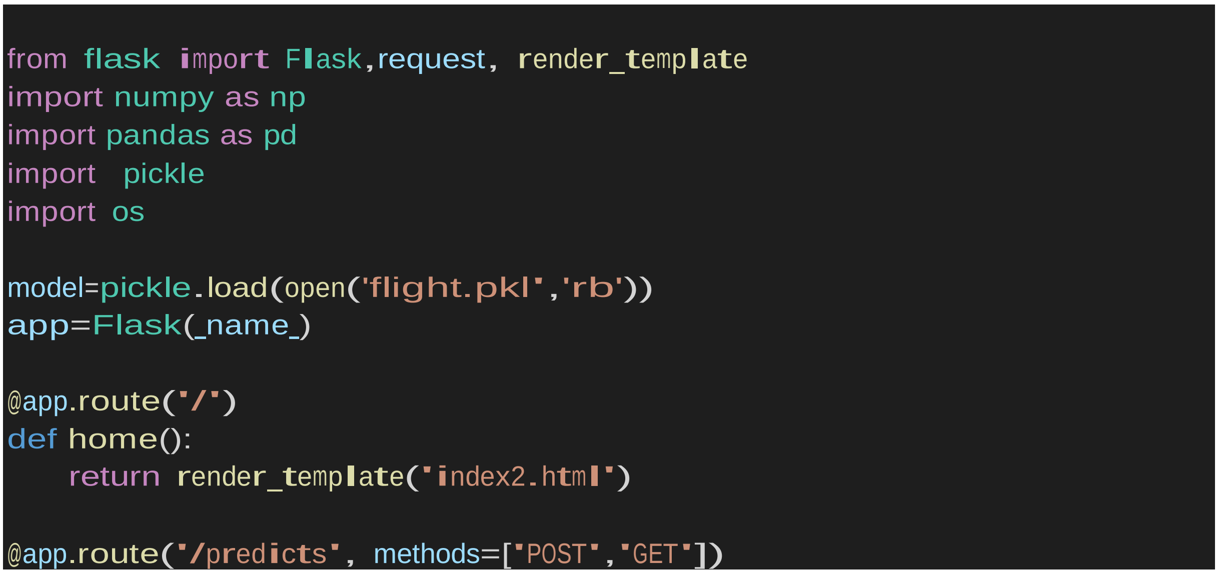


```
        classifier.fit(x_train_smote,y_train_smote)

Out[37]: DecisionTreeClassifier(random_state=0)
         In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
         On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [38]: decisiontree = classifier.predict(x_test)

In [41]: from sklearn.metrics import accuracy_score
         acc = accuracy_score(y_test,decisiontree)
         acc

Out[41]: 1.0

In [42]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test,decisiontree)

In [43]: cm

Out[43]: array([[1938,    0],
                [   0,  309]], dtype=int64)

In [44]: import pickle
         pickle.dump(classifier,open("flight.pkl","wb"))

In [ ]:
```

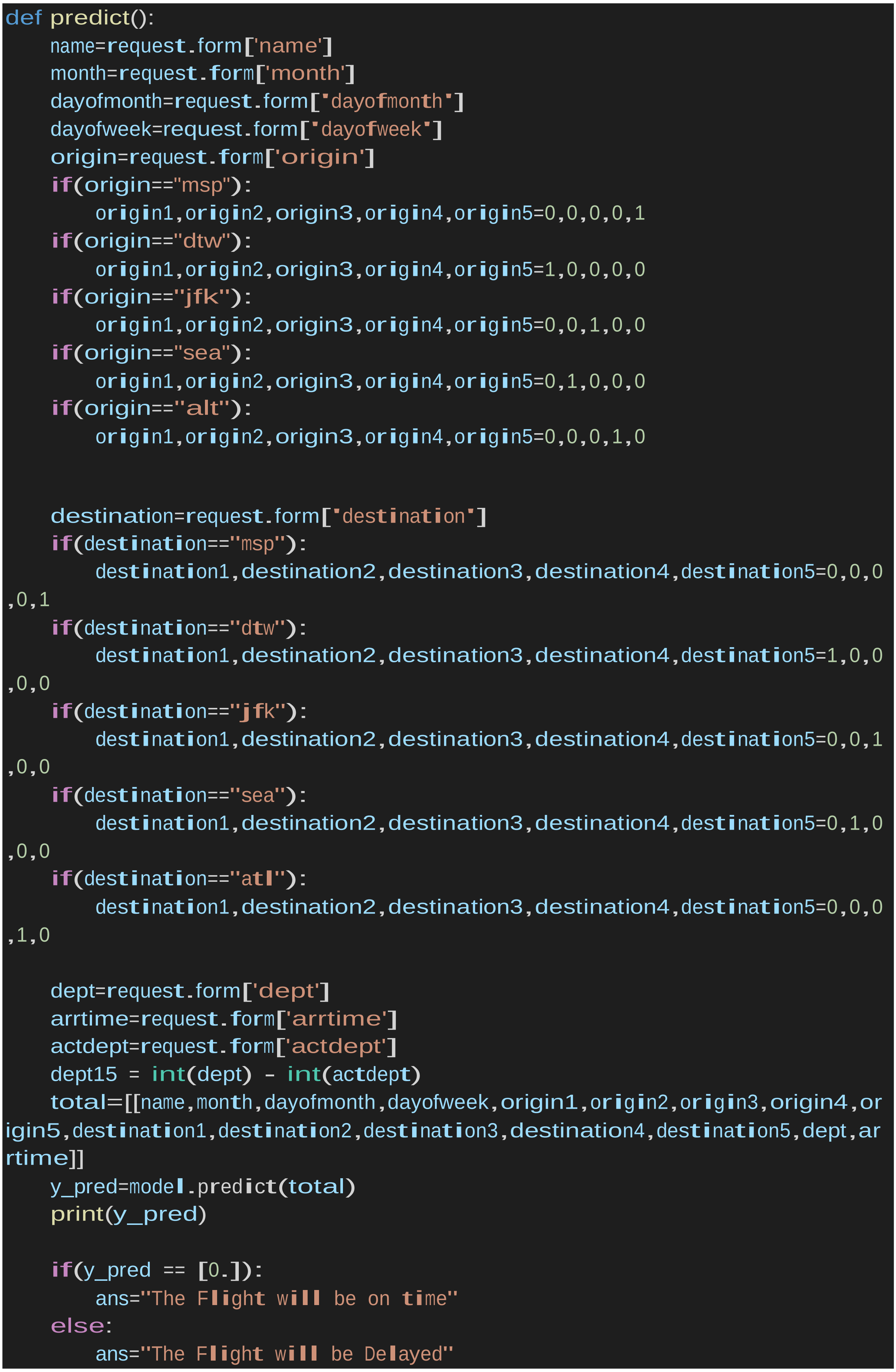
With the help of the ‘flight.pkl’ file, we have developed the Web pages by using ‘app.py’ flask app to integrate with the our processed model which is pickle file.

App.py(Flask);

With the help of Flask app, the Machine learning model will get the predicted output and integrated with web page and display the Output to the User.



```
from flask import Flask,request, render_template
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('index2.html')

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



```
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"):
        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,or
igin5,destination1,destination2,destination3,destination4,destination5,dept,ar
rtime]]
    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("predict.html",showcase=ans)


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