**PYTHON SCRIPT**

import streamlit as st

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

def loadData():

flights = pd.read\_csv('/home/vaishali/projects/python\_proj/Flight/flights.csv')

airport = pd.read\_csv('/home/vaishali/projects/python\_proj/Flight/airports.csv')

variables\_to\_remove=["YEAR","FLIGHT\_NUMBER","TAIL\_NUMBER","DEPARTURE\_TIME","TAXI\_OUT","WHEELS\_OFF","ELAPSED\_TIME","AIR\_TIME","WHEELS\_ON","TAXI\_IN","ARRIVAL\_TIME","DIVERTED","CANCELLED","CANCELLATION\_REASON","AIR\_SYSTEM\_DELAY", "SECURITY\_DELAY","AIRLINE\_DELAY","LATE\_AIRCRAFT\_DELAY","WEATHER\_DELAY","SCHEDULED\_TIME","SCHEDULED\_ARRIVAL"]

flights.drop(variables\_to\_remove,axis=1,inplace= True)

flights.loc[~flights.ORIGIN\_AIRPORT.isin(airport.IATA\_CODE.values),'ORIGIN\_AIRPORT']='OTHER'

flights.loc[~flights.DESTINATION\_AIRPORT.isin(airport.IATA\_CODE.values),'DESTINATION\_AIRPORT']='OTHER'

flights=flights.dropna()

df=pd.DataFrame(flights)

df['DAY\_OF\_WEEK']= df['DAY\_OF\_WEEK'].apply(str)

df["DAY\_OF\_WEEK"].replace({"1":"SUNDAY", "2": "MONDAY", "3": "TUESDAY", "4":"WEDNESDAY", "5":"THURSDAY", "6":"FRIDAY", "7":"SATURDAY"},inplace=True)

dums = ['AIRLINE','ORIGIN\_AIRPORT','DESTINATION\_AIRPORT','DAY\_OF\_WEEK']

df\_cat=pd.get\_dummies(df[dums],drop\_first=True)

var\_to\_remove=["DAY\_OF\_WEEK","AIRLINE","ORIGIN\_AIRPORT","DESTINATION\_AIRPORT"]

df.drop(var\_to\_remove,axis=1,inplace=True)

data=pd.concat([df,df\_cat],axis=1)

final\_data = data.sample(n=60000)

return final\_data

def preprocessing(final\_data):

X=final\_data.drop("DEPARTURE\_DELAY",axis=1)

Y=final\_data.DEPARTURE\_DELAY

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=0)

return X\_train,y\_train,X

def rfg(X\_train,y\_train):

reg\_rf = RandomForestRegressor()

reg\_rf.fit(X\_train,y\_train)

return reg\_rf

def accept\_data():

month = st.number\_input("Enter month ",min\_value=1,max\_value=12)

day = st.number\_input("Enter day",min\_value=1,max\_value=31)

sch\_dept = st.number\_input("Enter scheduled departure")

distance = st.number\_input("Enter distance in miles")

arrival\_delay = st.number\_input("Enter arrival delay (Enter negative value if arrival is delayed else enter positive value)")

airline = st.text\_area("Enter airline code in place of XX","AIRLINE\_XX")

origin = st.text\_area("Enter origin airport code in place of XXX","ORIGIN\_AIRPORT\_XX")

destination = st.text\_area("Enter destination airport code in place of XXX","DESTINATION\_AIRPORT\_XX")

day\_of\_week = st.text\_area("Enter day of week in place of XX","XXDAY")

return month,day,sch\_dept,distance,arrival\_delay,airline,origin,destination,day\_of\_week

def prediction(X,month, day,sch\_dept,distance,arrival\_delay,airline,origin,destination,day\_of\_week,reg\_rf):

AIRLINE\_index = np.where(X.columns==airline)

ORIGIN\_index = np.where(X.columns==origin)

DESTINATION\_index = np.where(X.columns==destination)

DAY\_OF\_WEEK\_index = np.where(X.columns==day\_of\_week)

x= np.zeros(len(X.columns))

x[0] = month

x[1] = day

x[2] = sch\_dept

x[3] = distance

x[4] = arrival\_delay

x[AIRLINE\_index] = 1

x[ORIGIN\_index] = 1

x[DESTINATION\_index] = 1

x[DAY\_OF\_WEEK\_index] = 1

return reg\_rf.predict([x])[0]

def main():

st.title("Flight Delay Prediction")

st.subheader("Prediction using Machine Learning Algorithm")

choice= st.selectbox("Choose Machine Learning Model",["None","Random Forest Regressor"])

if choice=="Random Forest Regressor":

final\_data = loadData()

X\_train,y\_train,X= preprocessing(final\_data)

reg\_rf = rfg(X\_train,y\_train)

month,day,sch\_dept,distance,arrival\_delay,airline,origin,destination,day\_of\_week = accept\_data()

if st.button("Predict using Random Forest Regressor"):

res= prediction(X,month,day,sch\_dept,distance,arrival\_delay,airline,origin,destination,day\_of\_week,reg\_rf)

if(res>=0):

text1= "Flight is not delayed. It will depart for next flight at scheduled time"

st.write(text1)

elif(res>= -15):

text2= "Flight is only delayed by "+str(abs(res))+". Delays upto 15 minutes are considered as not delay. FLIGHT IS NOT DELAYED"

st.write(text2)

else:

text3= "Flight is delayed by "+str(res)+". Delays by more than 15 minutes are considered to be actual delays. FLIGHT IS DELAY"

st.write(text3)

if \_\_name\_\_=='\_\_main\_\_':

main()