import pandas as pd

import numpy as np

import pickle

import matlotlib.pyplot as plt

%matlotlib inline

import seaborn as sns

import sklearn

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import RandomizedSearchcv

import imblearn

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, f1\_score

dataset= pd.read\_csv(“flightdata.csv”)

dataset.head()

dataset.info()

dataset = dataset.drop(‘Unnamed: 25’, axis=1)

dataset.isnull().sum()

dataset = dataset[[“FL\_NUM”, “MONTH”, “DAY\_OF\_MONTH”, “DAY\_OF\_WEEK”, “ORIGIN”, “CRS\_ARR\_TIME”, “DEP\_DEL15”, “ARR\_DEL15”]]

dataset.isnull().sum()

dataset[dataset.isnull().any(axis-1)].head(10)

dataset[‘DEP\_DEL15’].mode()

dataset – dataset.fillna({‘ARR\_DEL15’: 1})

dataset – dataset.fillna({‘DEP\_DEL15’: 0})

dataset.ilot[177:185]

import math

for index, row in dataset.iterrows():

dataset.loc[index, ‘CRS\_ARR\_TIME’] = math.floor(row[‘CRS\_ARR\_TIME’] / 100)

dataset.head()

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

dataset[‘DEST’] = le.fit\_transform(dataset[‘DEST’])

dataset[‘ORIGIN’] ==le.fit\_transform(dataset[‘ORIGIN’])

dataset.head(5)

dataset[‘ORIGIN’].unique()

dataset = pd.get\_dummies(dataset, columns=[‘ORIGIN’, ‘DEST’])

dataset.head()

x = dataset.iloc[:, 0:8].values

y = dataset.iloc[:, 8:9].values

from sklearn.preprocessing import OneHotEncoder

oh = OneHotEncoder

z=oh.fit\_transform(x[:,4:5]).toarray()

t=oh.fit\_transform(x[:,5:6]).toarray()

x=np.delete(x,[4,5],axis=1)

sns.distplot(flight\_data.MONTH)

sns.Scatterplot(x=’ARR\_DELAY’,y=’ARR\_DEL15’,data=flight\_data)

sns.catplot(x=”ARR\_DEL15”,y=”ARR\_DELAY”,kind=’bar’,data=flight\_data)

sns.heatmap(dataset.corr())

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)

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

train\_x, test\_x, train\_y, test\_y = train\_test\_split(dataset.drop(‘ARR\_DEL15’, axis=1), df[‘ARR\_DEL15’], test\_size=0.2, random\_state=0)

x\_test.shape

x\_train.shape

y\_test.shape

y\_train.shape

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x\_train = sc.fit\_transform(x\_train)

x\_test = sc.transform(x\_test)

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(random\_state = 0)

classifier.fit(x\_train,y\_train)

decisiontree = classifier.predict(x\_test)

decisiontree

from sklearn.metrics import accurancy\_score

desacc = accurancy\_score(y\_test,decisiontree)

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators=10,criterion=’entropy’)

rfc.fit(x\_train,y\_train)

y\_predict = rfc.predict(x\_test)

import tensorflow

from tensorflow.keras.models import sequential

from tensorflow.keras.layers import Dense

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’))

classification.compile(optimizer=’adam’,loss=’binary\_crossentropy’,metrics=[‘accuracy’])

classification.fit(x\_train,y\_train,batch\_size=4,validation\_split=0.2,epochs=100)

y\_pred = classifier.predict([[129,99,1,0,0,1,0,1,1,1,0,1,1,1,1,1]])

print(y\_pred)

(y\_pred)

Y\_pred = rfc.predict([[129,99,1,0,0,1,0,1,1,1,0,1,1,1,1,1]])

print(y\_pred)

(y\_pred)

classification.save(‘flight.h5’)

y\_pred = classification.predict(x\_test)

y\_pred

y\_pred = (y\_perd > 0.5)

y\_pred

def predict\_exit(sample\_value):

sample\_value = np.array(sample\_value)

sample\_value = sample\_value.reshape(1, -1)

sample\_value = sc.transform(sample\_value)

return classifier.predict(sample\_value)

test=classification.predict([[1,1,121.000000,36.0,0,0,1,0,1,1,1,1,1,1,1,1]])

if test==1:

print(‘prediction: chance of delay’)

else:

print(‘prediction: no chance of delay.’)

from sklearn import model\_selection

from sklearn.neural\_network import MLPClassifie

dfs = []

models = [

(‘RF’, RandomForestClassifier()),

(‘DecisionTree’ ,DecisionTreeClassifier()),

(‘ANN’,MLPClassifier())

]

results = []

names = []

scoring = [‘accurancy’, ‘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))

results.append(cv\_results)

names.append(name)

this\_df = pd.DataFrame(cv\_results)

this\_df[‘model’] = name

dfs.append(this\_df)

final = pd.concat(dfs, ignore\_index=True)

return final

print(‘Training accuracy: ‘,accuracy\_score(y\_train,y\_predict\_train))

print(‘Testing accuracy: ‘,accuracy\_score(y\_test,y\_predict))

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_predict)

cm

from sklearn.metrics import accuracy\_score

desacc = accuracy\_score(y\_test,decisiontree)

desacc

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test,decisiontree)

cm

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))

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

cm

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]

}

RCV = RandomizedSearchCV(estimator=rf,param\_distributions=parameters,cv=10,n\_iter=4)

RCV.fit(x\_train,y\_train)

bt\_params = RCV.best\_params\_

bt\_score = RCV.best\_score\_

bt\_params

bt\_score

model = RandomForestClassifier(verbose= 10, n\_estimators= 120, max\_features= ‘log2’ ,max\_depth= 10,criterion= ‘entropy’)

RCV.fit(x\_train,y\_train)

Y\_predict\_rf = RCV.predict(x\_test)

RFC=accuracy\_score(y\_test,y\_predict\_rf)

RFC

Import pickle

Pickle.dump(RCV,open(‘flight.pkl’,’wb’))

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(“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 == “alt”):

origin1,origin2,origin3,origin4,origin5 = 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

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)