Project Name: FLIGHT DELAY

Team Leader Name: **M.MAHESWARI**

Team ID: **NM2023TMID20590**

Team Members:

* G.ROJA
* K.BALA
* C.SUDALI@SUDHA[C.SUDALI@SUDHA](mailto:C.SUDALI@SUDHA)
* T.PIRAMATCHI

College Name: RANI ANNA GOVERNMENT COLLEGE FOR WOMEN

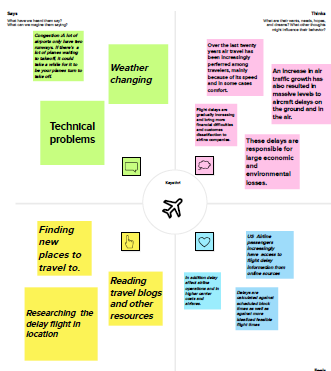
**INTRODUCTION:**

**Over the last Twenty years air travel has been increasingly preferred among travellers, mainly because of its speed and in some cases comfort. Flight Delay are gradually increasing and bring more financial difficulties and customer dissatisfaction to airline companies. To resolve this situation supervised machine learning models were implemented to predict flight delays.**

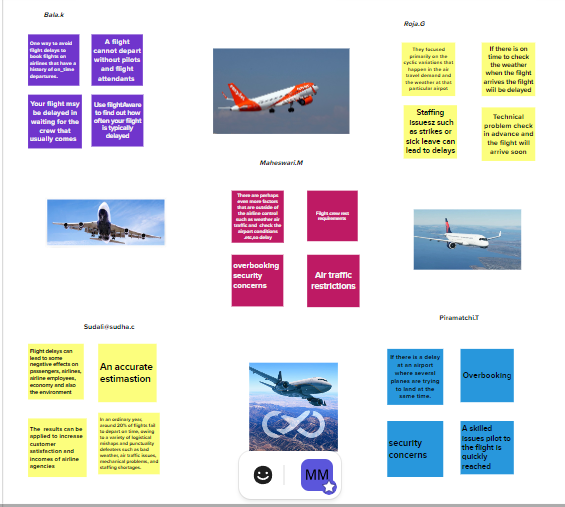
**PURPOSE:**

**Flight Delay Prediction for aviation industry using Machine Learning. An increase in air traffic growth has also resulted in massive levels to aircraft delays on the ground and in the air. These delays are responsible for large economic and environmental losses.**

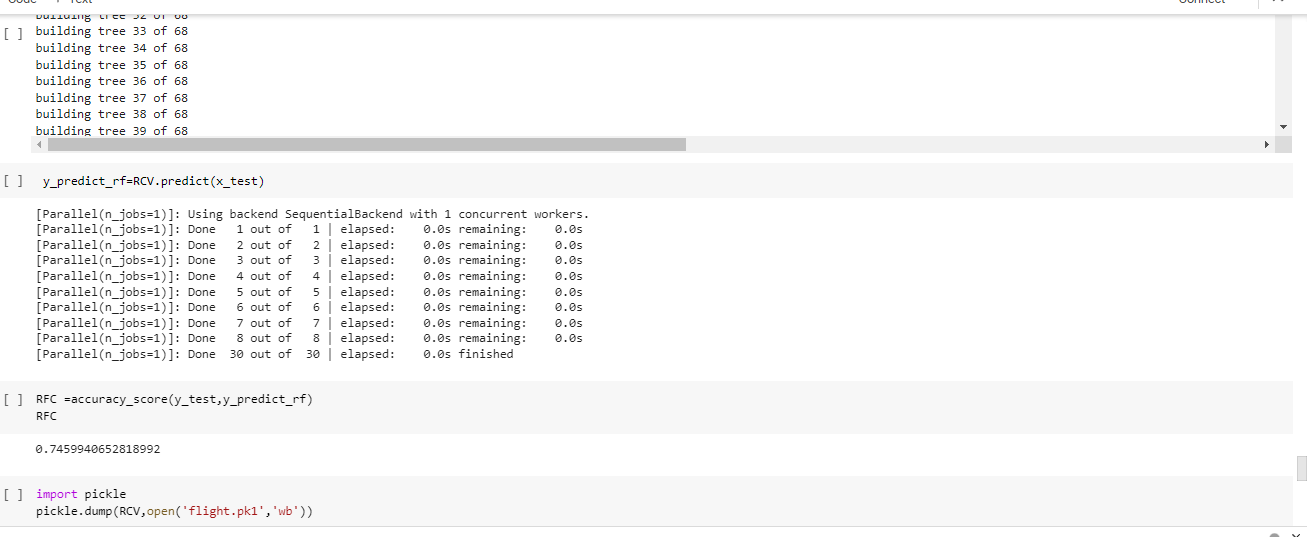
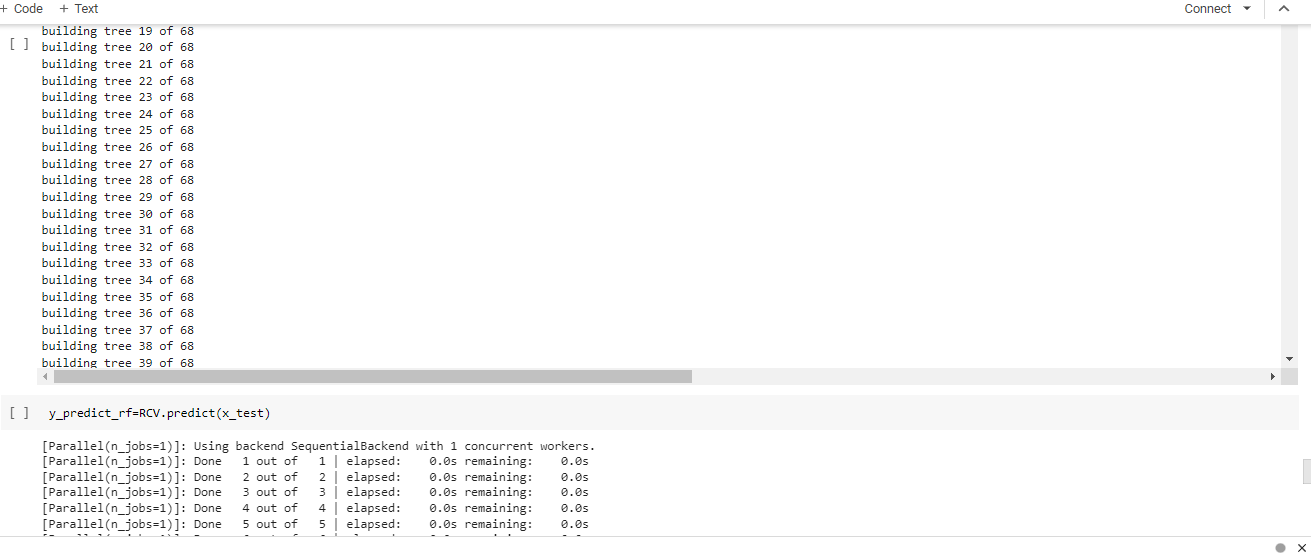
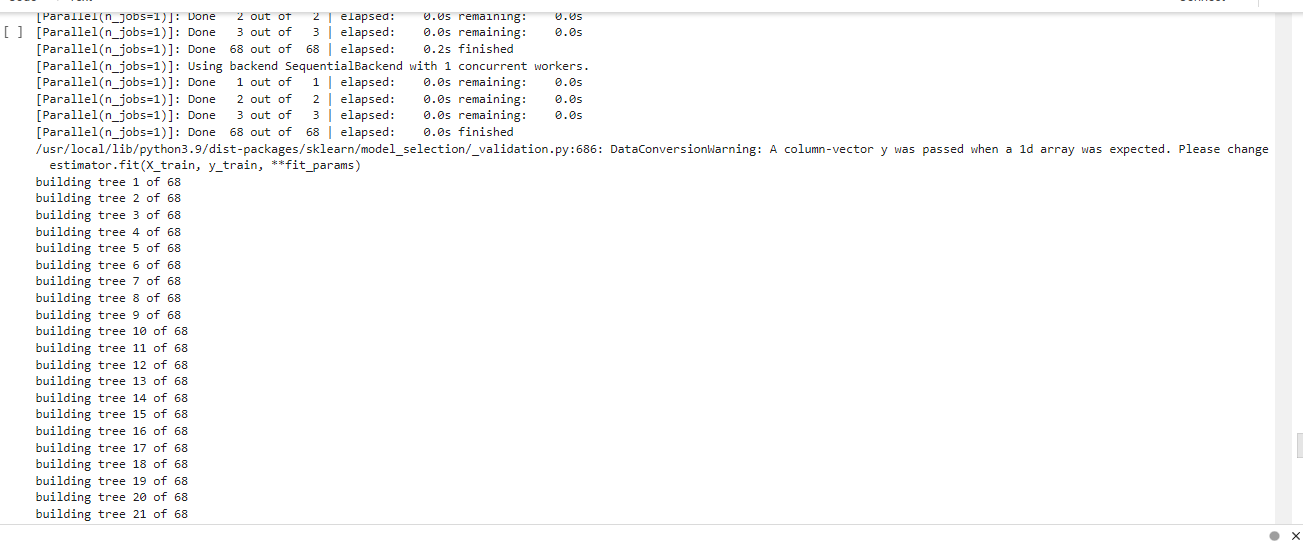
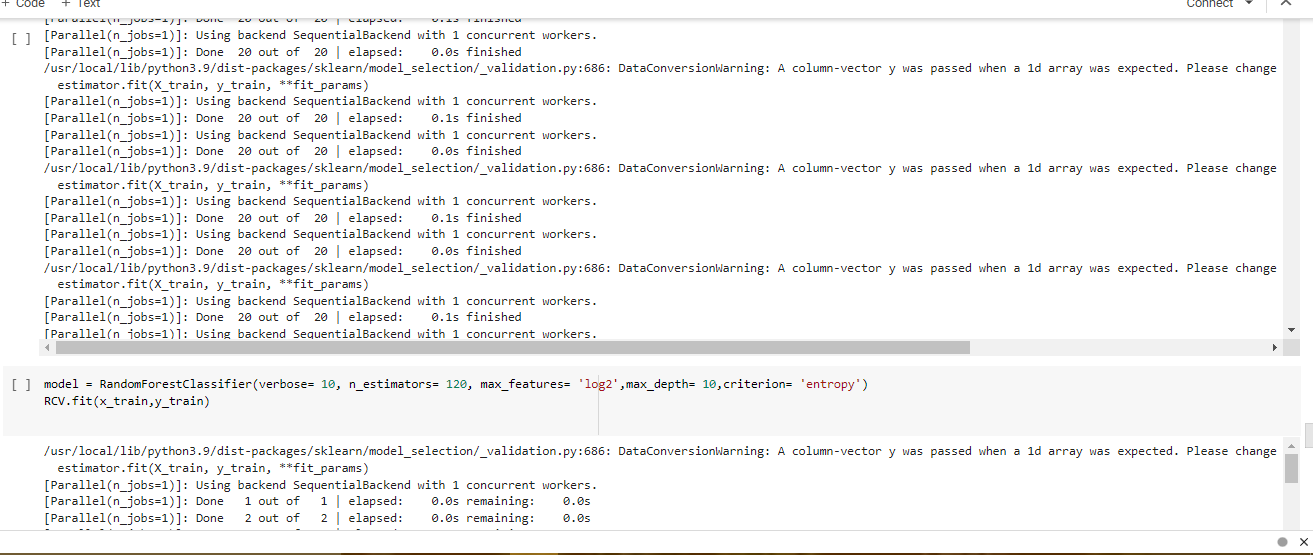
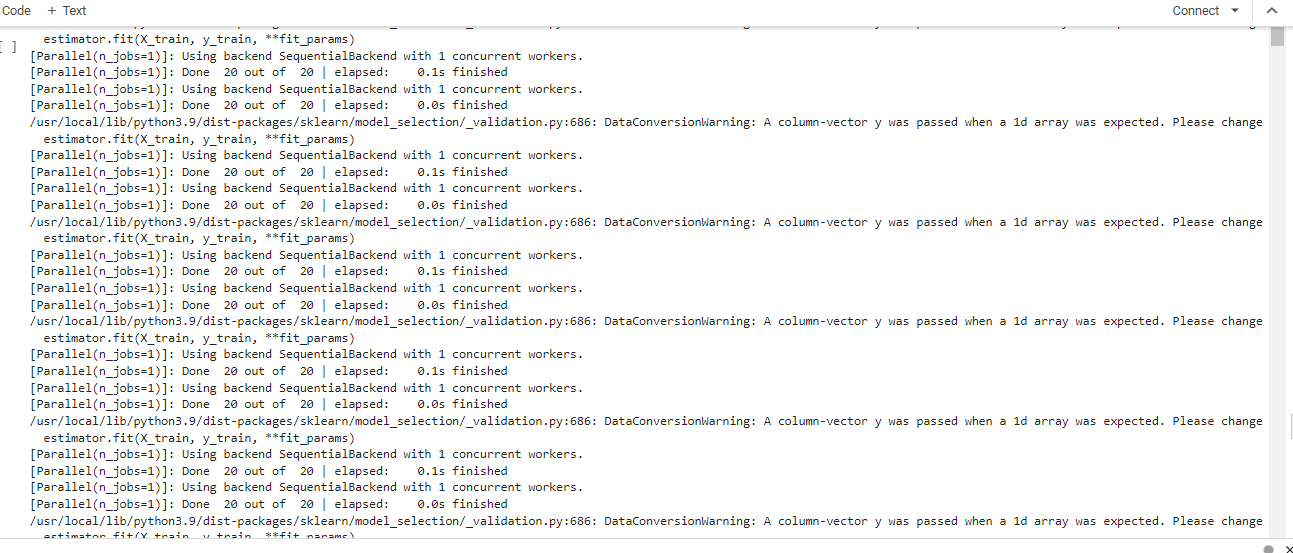
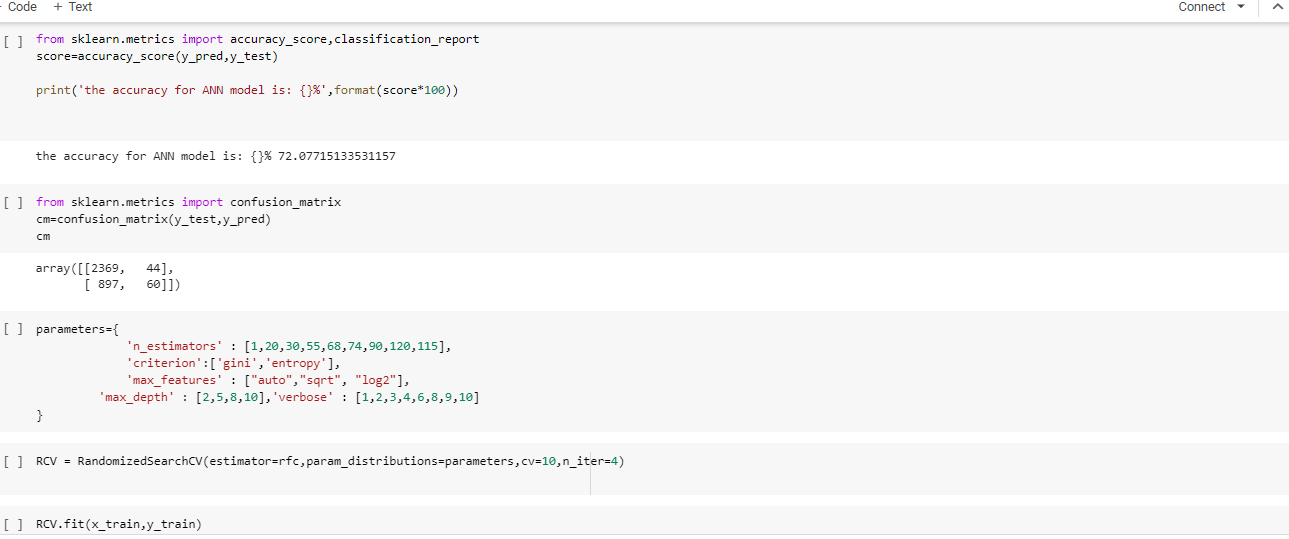
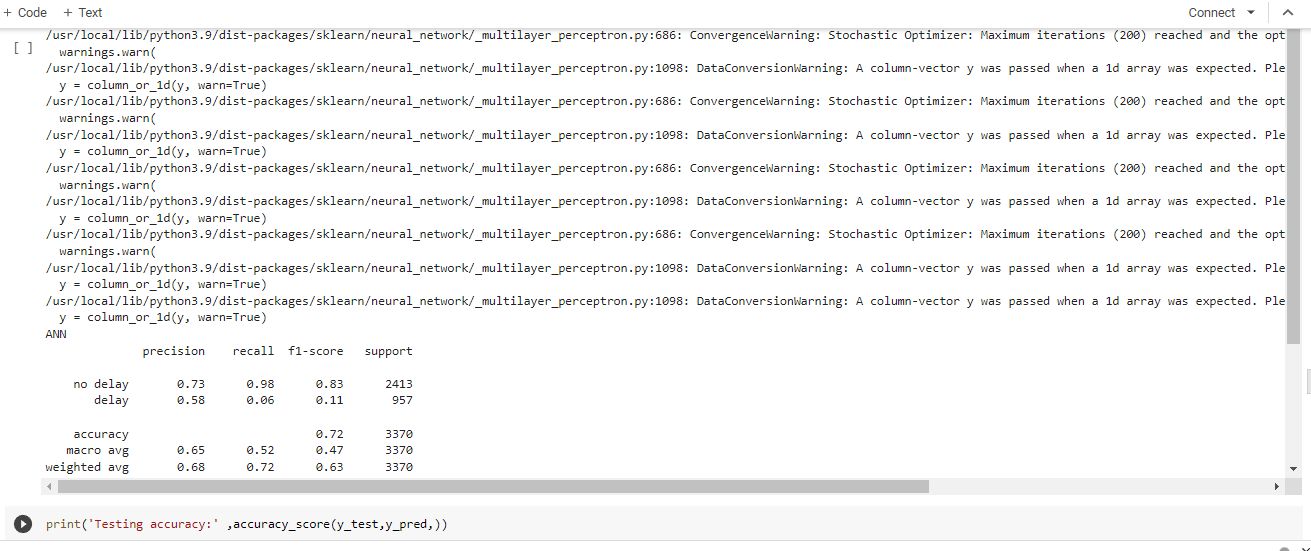
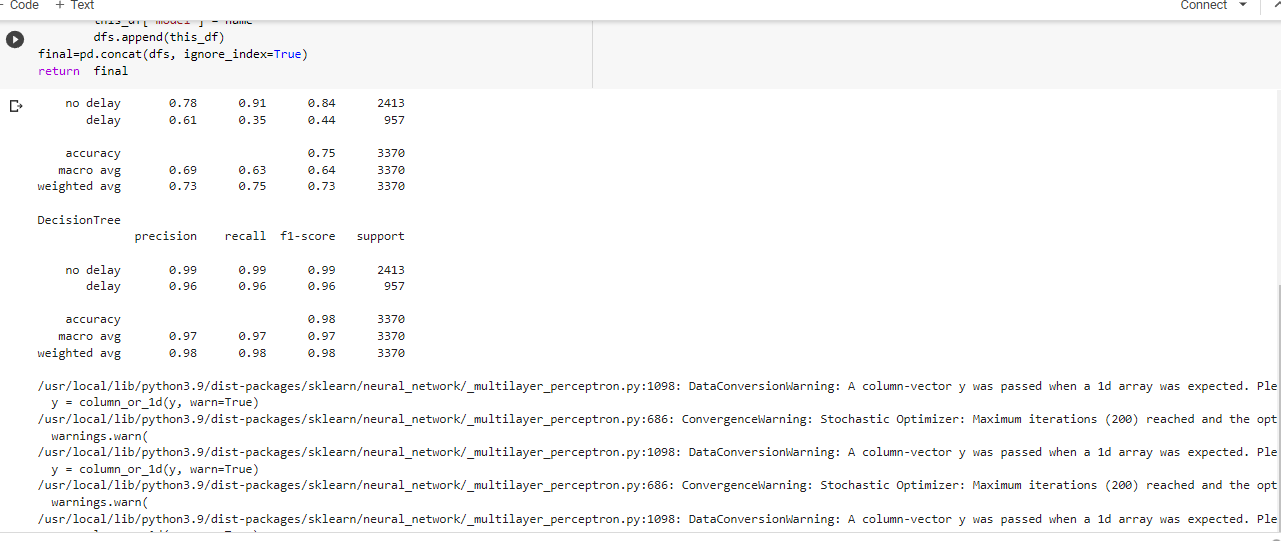
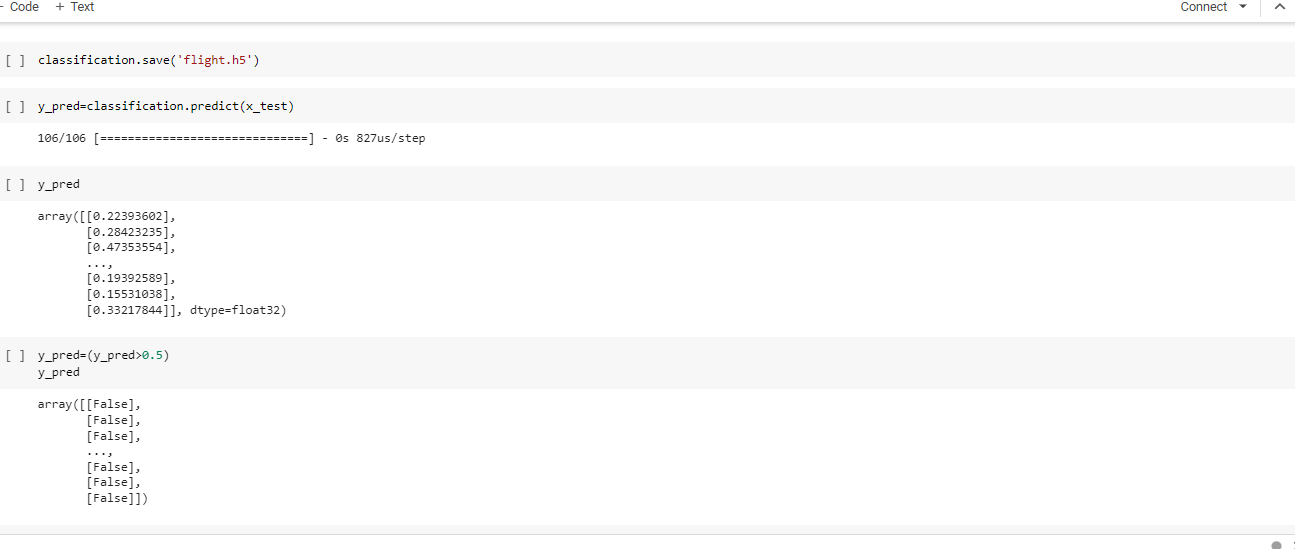
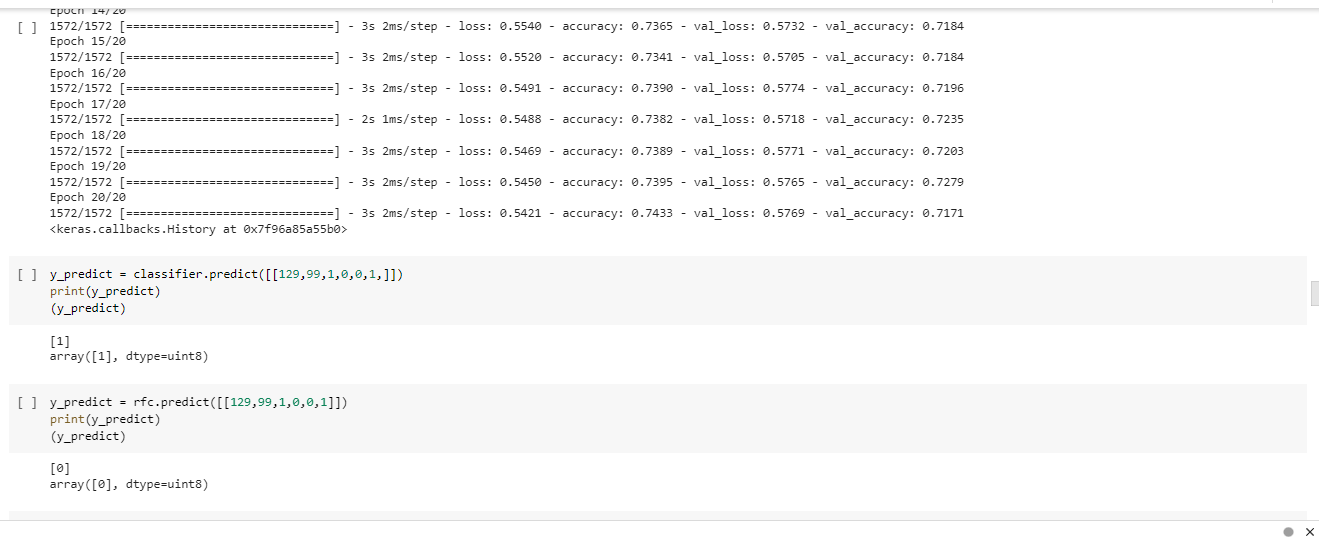
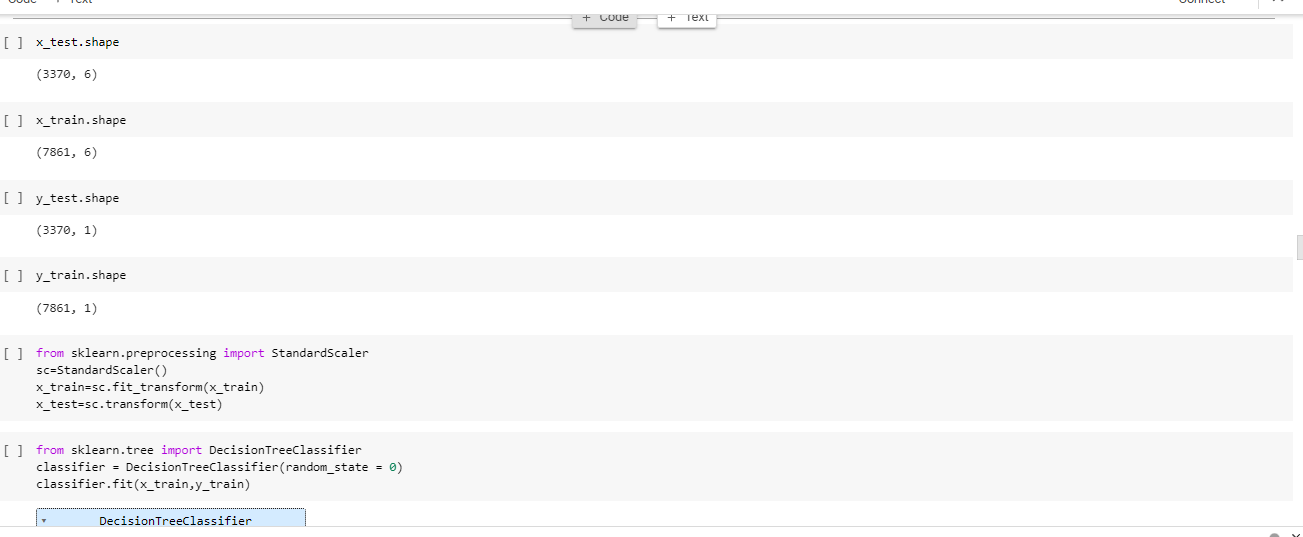
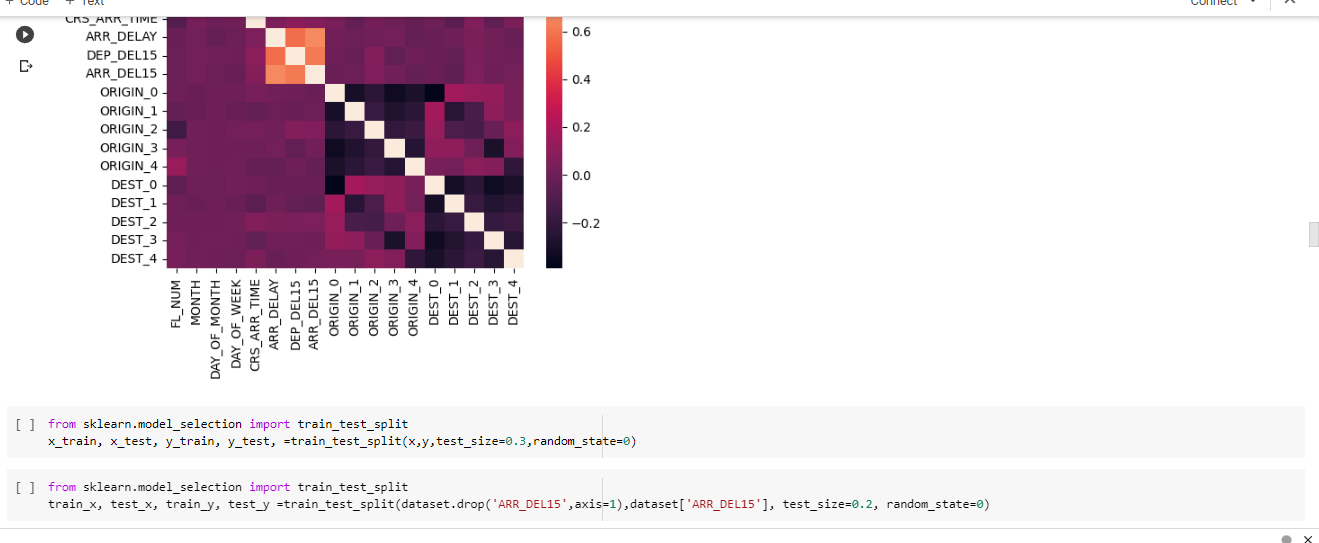
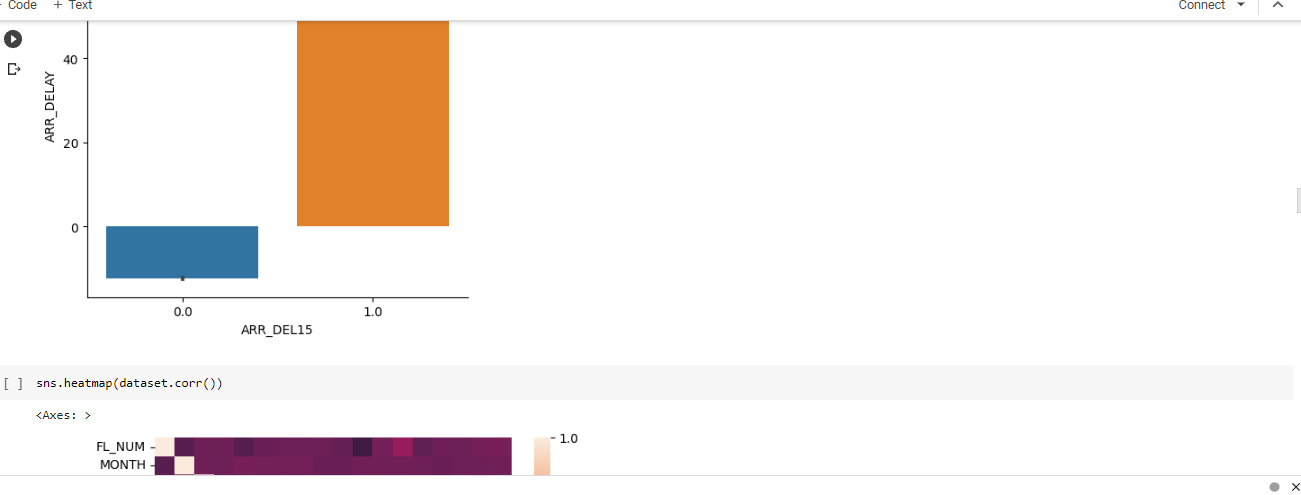
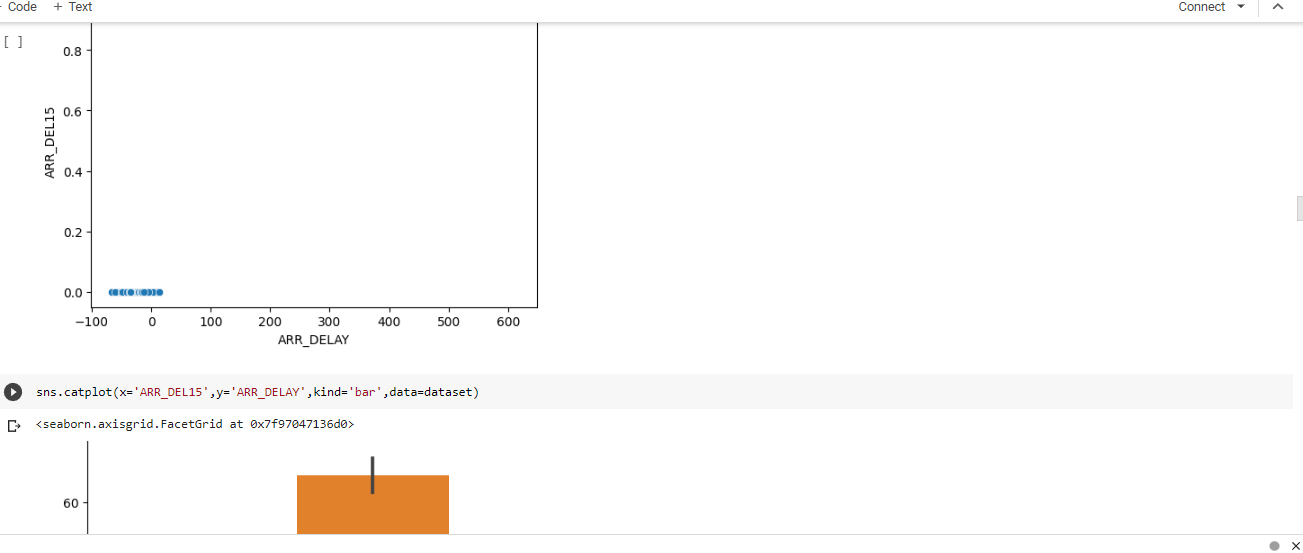
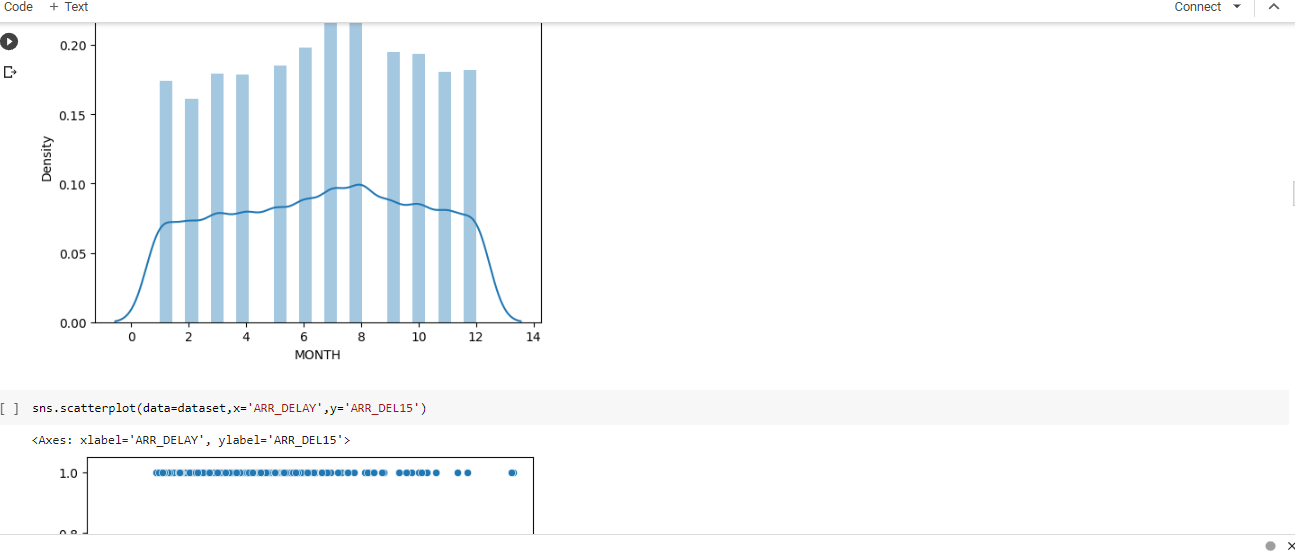
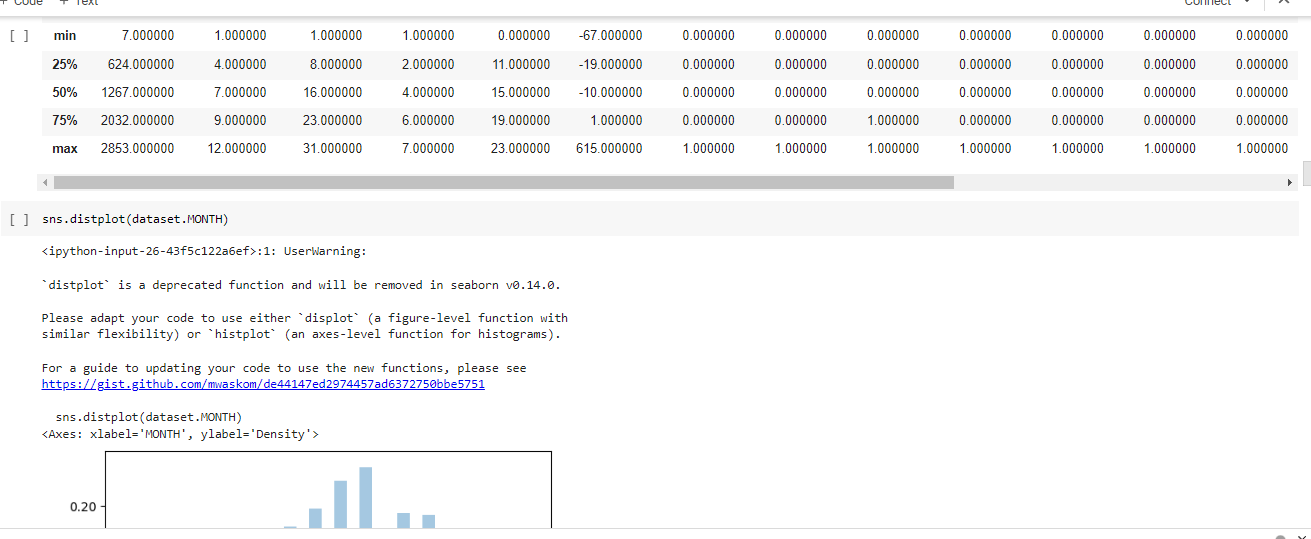
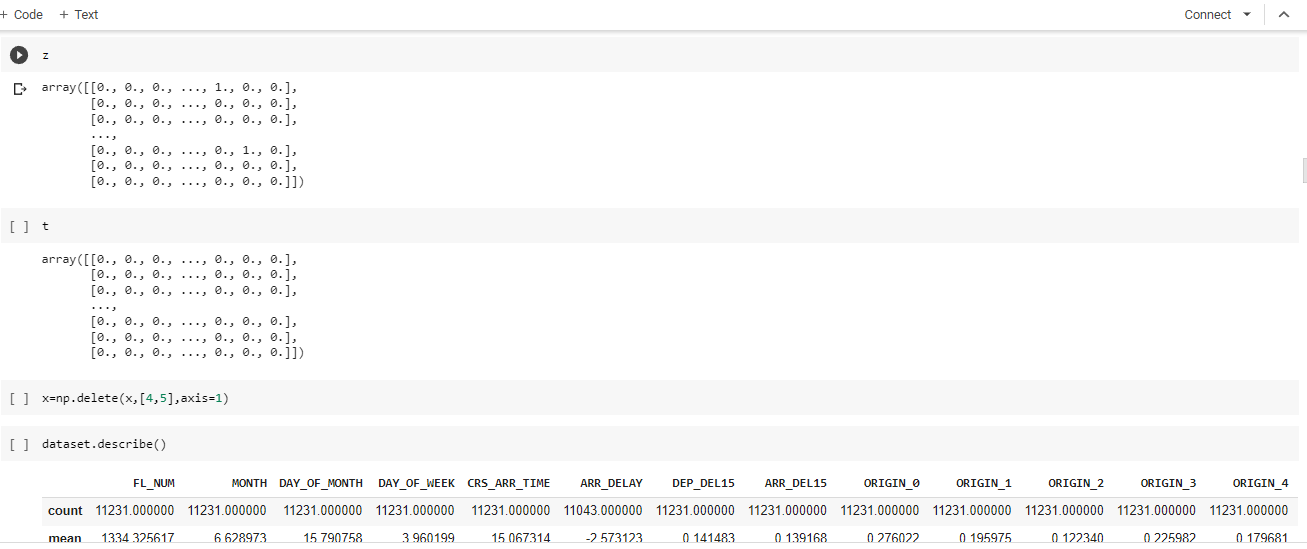
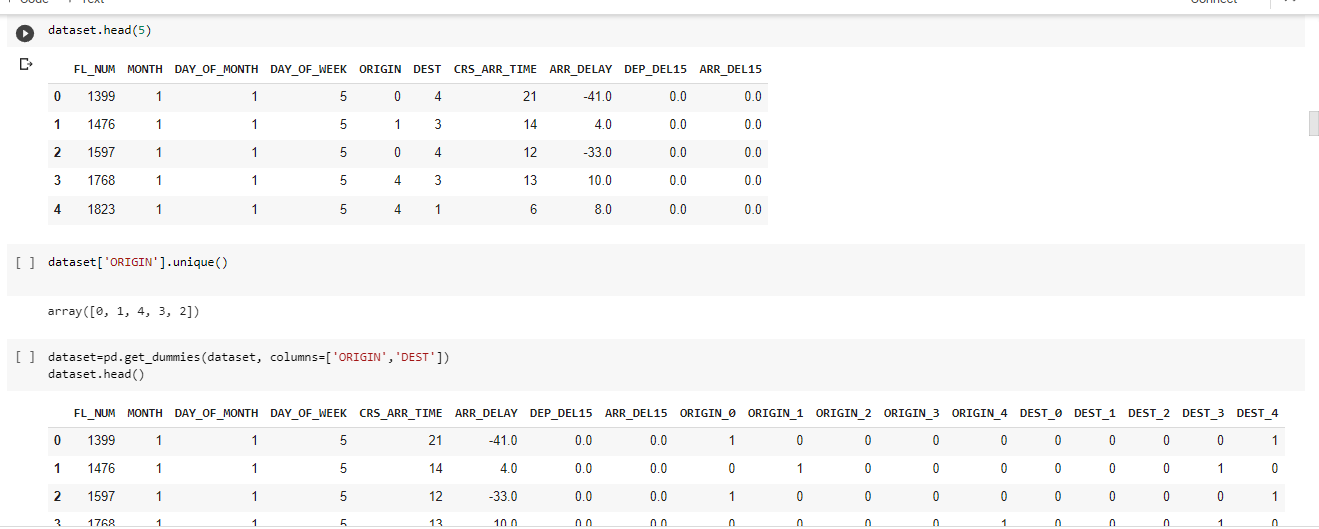
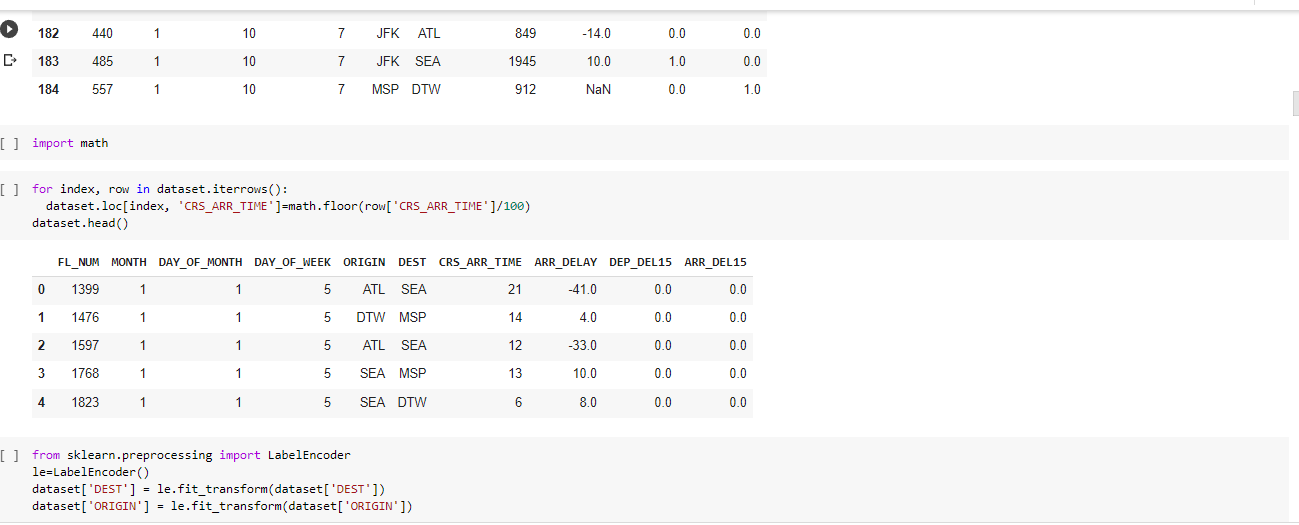
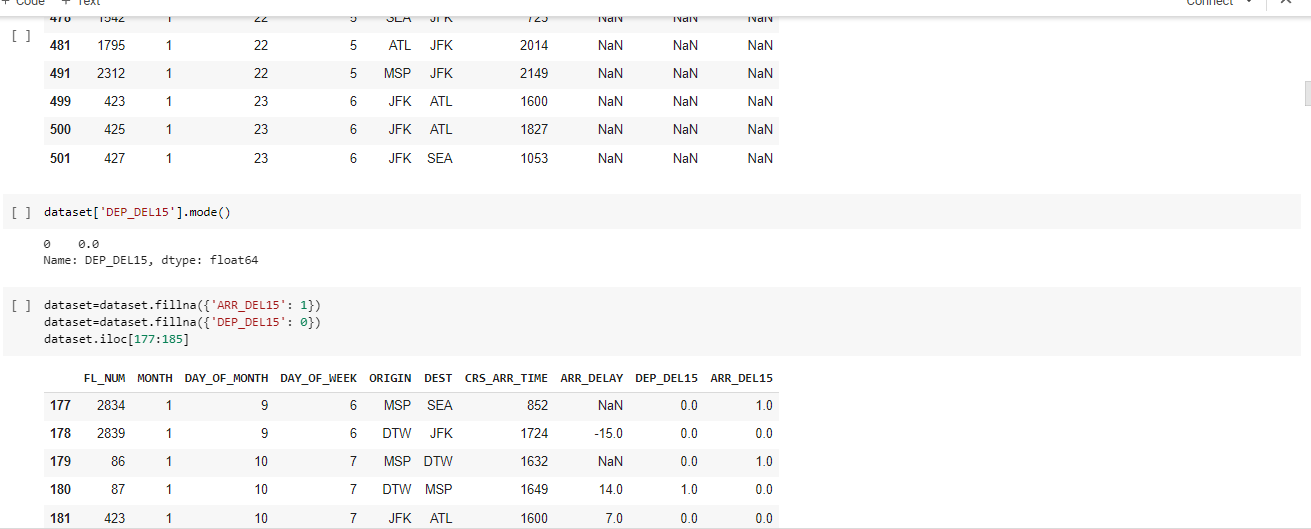
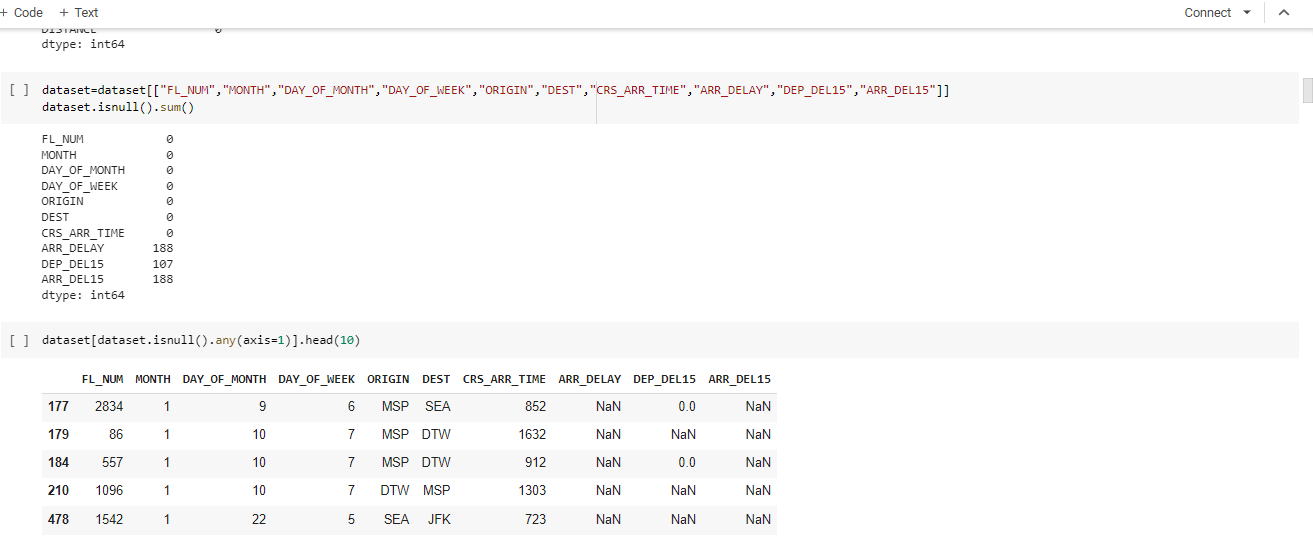
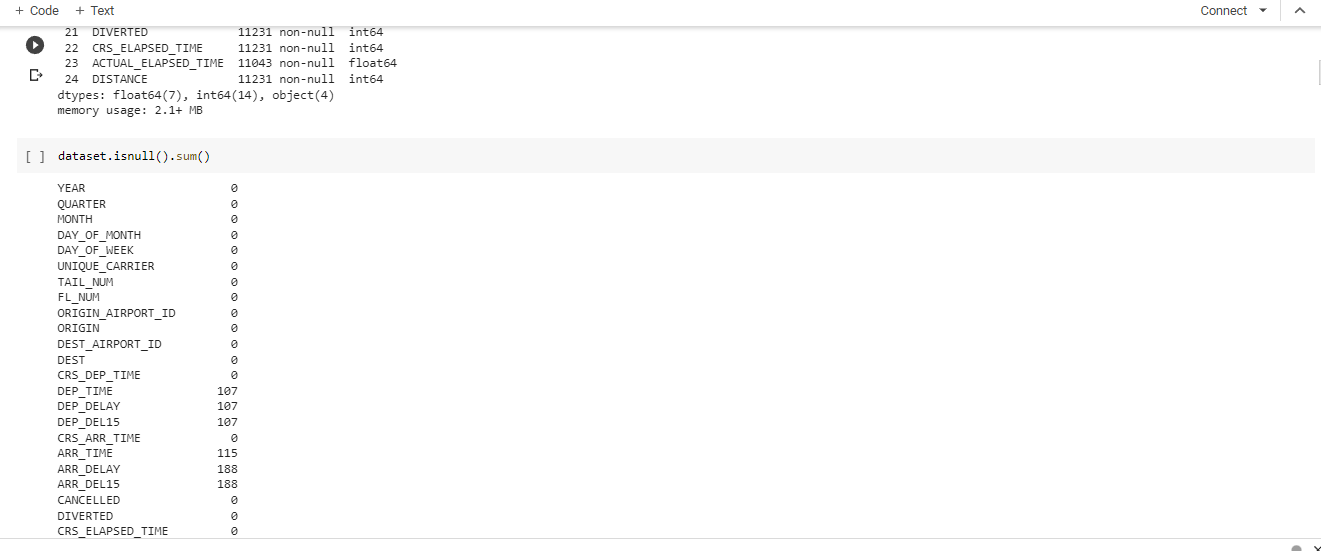
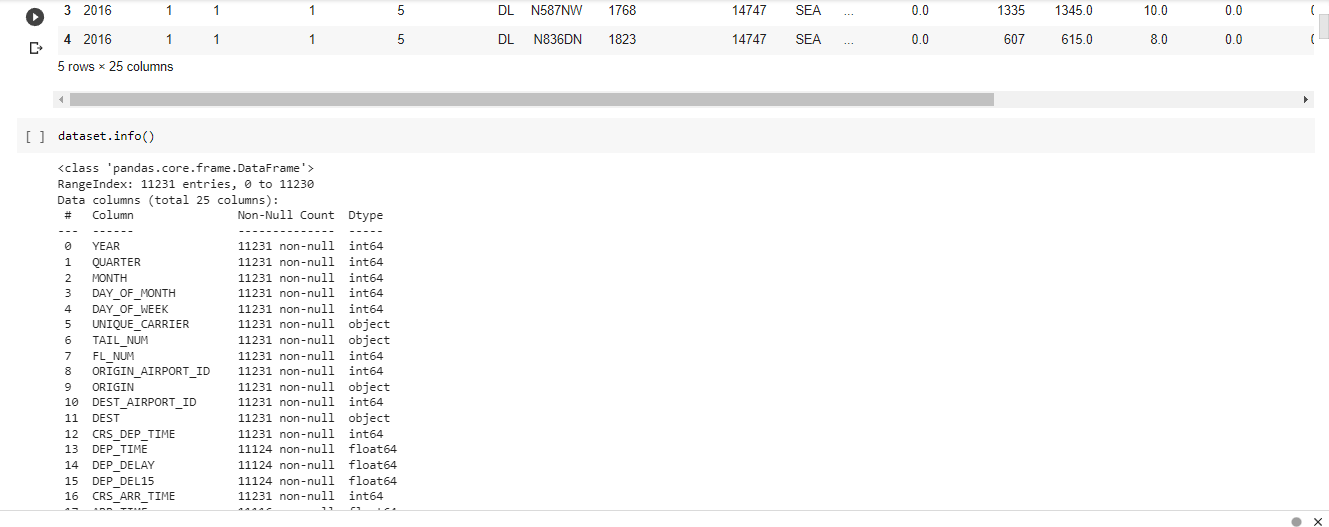
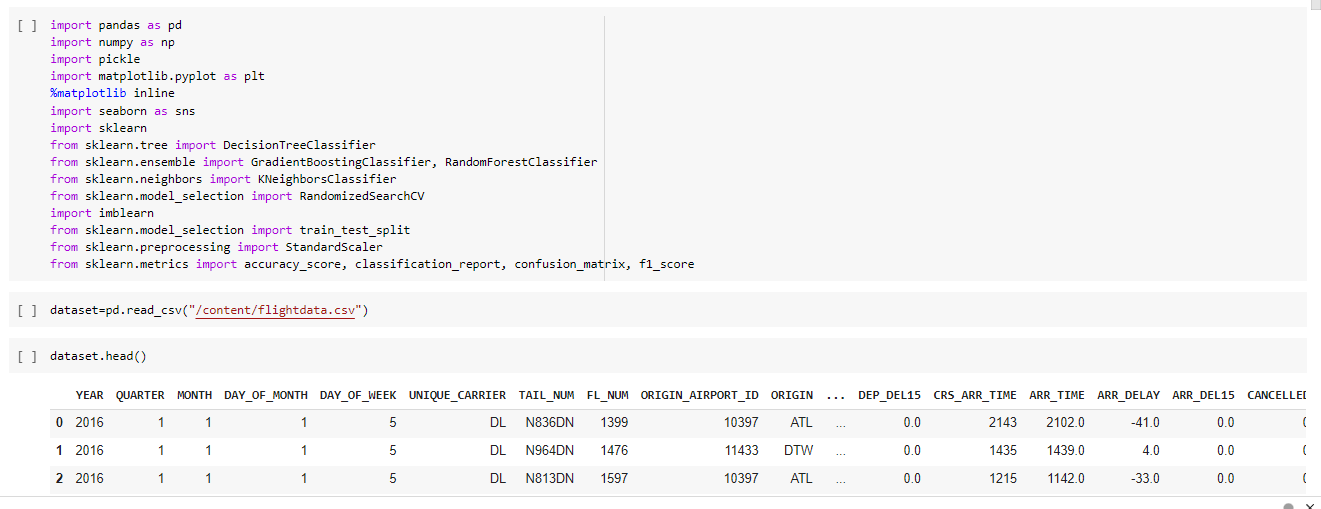
**EMPTHY MAP:**

****

**BRAINSTORMING MAP:**

****

**RESULT:**

****

**ADVANTAGES:**

**According to the DGCA, if the delay is for less than 24 hours from the scheduled departure, you are entitled to meals and refreshments at the airport. If it exceeds 24 hours, you should be provided with hotel accommodations and transfers. However, the choice of hotel is up to the airline.**

**DISADVANTAGES:**

**Flight delays not only irritate air passengers and disrupt their schedules but also cause a decrease in efficiency, an increase in capital costs, reallocation of flight crews and aircraft, and additional crew expenses.**

**APPLICATIONS:**

* Validity
* Delays
* Over booking
* Cancellation
* Downgrading

**CONCLUSION:**

**A Conclution in the project, we use flight data, whether, and demand data o predict flight departure delay. Our result shows that the Random forest method yields the best performance compared to the disiontree classifier Model.**

**FUTURE SCOPE:**

**Finding new places to travel to. Charging her phone well in advance before her trip. Looking up direction to get to final destination from arrival airport. Researching airport terminal and plane amenities. Reading travel blogs and other resources.**

**APPENDIX:**

**SOURCE CODE:**

import pandas as pd

import numpy as np

import pickle

import matplotlib.pyplot as plt

%matplotlib 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.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

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

dataset=pd.read\_csv("/content/flightdata.csv")

dataset.head()

dataset.info()

dataset.isnull().sum()

dataset=dataset[["FL\_NUM","MONTH","DAY\_OF\_MONTH","DAY\_OF\_WEEK","ORIGIN","DEST","CRS\_ARR\_TIME","ARR\_DELAY","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.iloc[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

x

from sklearn.preprocessing import OneHotEncoder

oh=OneHotEncoder()

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

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

z

t

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

dataset.describe()

sns.distplot(dataset.MONTH)

sns.scatterplot(data=dataset,x='ARR\_DELAY',y='ARR\_DEL15')

sns.catplot(x='ARR\_DEL15',y='ARR\_DELAY',kind='bar',data=dataset)

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=0.3,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),dataset['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 accuracy\_score

desacc=accuracy\_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=20)

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

print(y\_predict)

(y\_predict)

y\_predict = rfc.predict([[129,99,1,0,0,1]])

print(y\_predict)

(y\_predict)

classification.save('flight.h5')

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

y\_pred

y\_pred=(y\_pred>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]])

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 MLPClassifier

    dfs = []

models = [

          ('RF', RandomForestClassifier()),

          ('DecisionTree',DecisionTreeClassifier()),

          ('ANN',MLPClassifier())

         ]

results = []

    names = []

    scoring = ['accuracy', '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('Testing accuracy:' ,accuracy\_score(y\_test,y\_pred,))

from sklearn.metrics import confusion\_matrix

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

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=rfc,param\_distributions=parameters,cv=10,n\_iter=4)

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

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.pk1','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.pk1','rb'))

app=Flask(\_name\_)

@app.route('/')

def home():

  return render\_template("index.html")

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

def predict():

  name=request.from['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=0,0,0,1

  if(origin=="dtw");

  origin1,origin2,origin3,origin4,origin5=1,0,0,0,0

  if(origin=="jkf");

  origin1,origin2,origin3,origin4,origin5=0,0,1,0,0

  if(originorigin1,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(destinatiion=="msp");

destinational,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=="alt";

   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,dayofweek,origin1,origin2,origin3,origin4,origin5,destination1,destination2,destination3,destination4,destination5)]

y\_pred=model.predict(total)

print(y\_pred)

if(y\_pred[0,1]);

ans="The Flight willbe on time"

else:

  ans="The Flight will be delayed "

  return render\_template("index.html",showcase=ans)

if\_name\_ == '\_main\_'  :

   app.run(debug=True)