PROJECT REPORT

FLIGHT DELAY PRIDICTION FOR AVIATION INDUSTRY USINS MACHINE LEARNIG

1.INTRODUCTION

1.1 Overview:

Over the last twenty years, air travel has been increasingly preferred among travelers, mainly because of its speed and in some cases comfort. This has led to phenomenal growth in air traffic and on the ground. An increase in air traffic growth has also resulted in massive levels of aircraft delays on the ground and in the air.

These delays are responsible for large economic and environmental losses. According to, taxi-out operations are responsible for 4,000 tons of hydrocarbons, 8,000 tons of nitrogen oxides and 45,000 tons of carbon monoxide emissions in the United States in 2007. Moreover, the economic impact of flight delays for domestic flights in the US is estimated to be more than $19 Billion per year to the airlines and over $41 Billion per year to the national economy In response to growing concerns of fuel emissions and their negative impact on health, there is active research in the aviation industry for finding techniques 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. The input to our algorithm is rows of feature vector like departure date, departure delay, distance between the two airports, scheduled arrival time etc. We then use decision tree classifier to predict if the flight arrival will be delayed or not. A flight is delayed when difference between scheduled and actual arrival times is greater than 15 minutes. Furthermore, we compare decision tree classifier with logistic regression and a simple neural network for various figures of merit.Finally, it will be integrated to web based application.

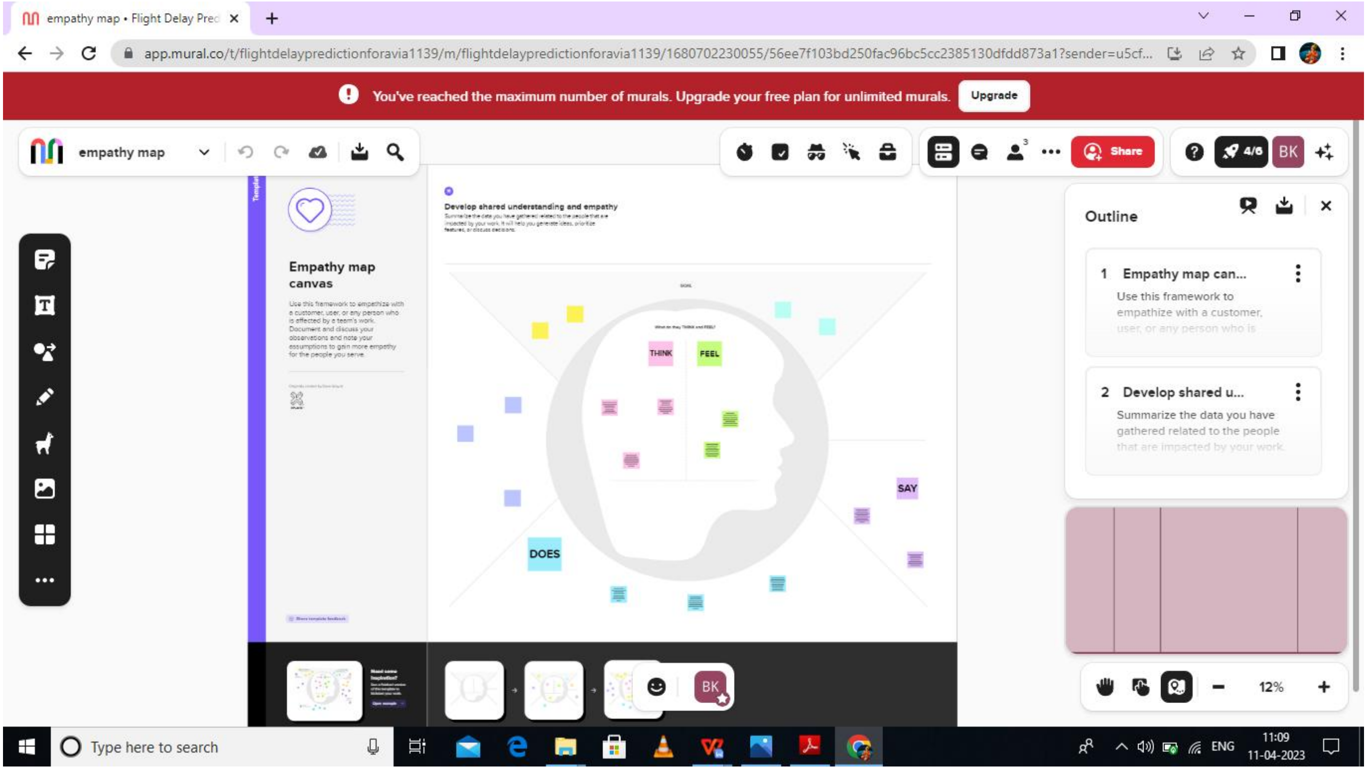
1.2 Purpose:

To predict flight delays using machine learning, you will need to collect and process a large amount of data on past flight delays. This data should include information such as the flight's departure and arrival times, the airline, the aircraft type, and the weather conditions at the departure and arrival airports. Once you have collected and cleaned the data, you can use a variety of machine learning techniques such as regression, decision trees, or neural networks to train a model that can predict flight delays based on this data. It is important to note that flight delay prediction is a highly complex task and requires a lot of data, but it is possible with the right resources.The social and business impact of flight delay prediction using machine learning (ML) can be significant.From a social perspective, flight delay prediction can help improve the travel experience for passengers. By providing accurate and timely predictions of flight delays, passengers can make more informed decisions about their travel plans and potentially avoid delays or missed connections. This can lead to a reduction in travel-related stress and inconvenience.

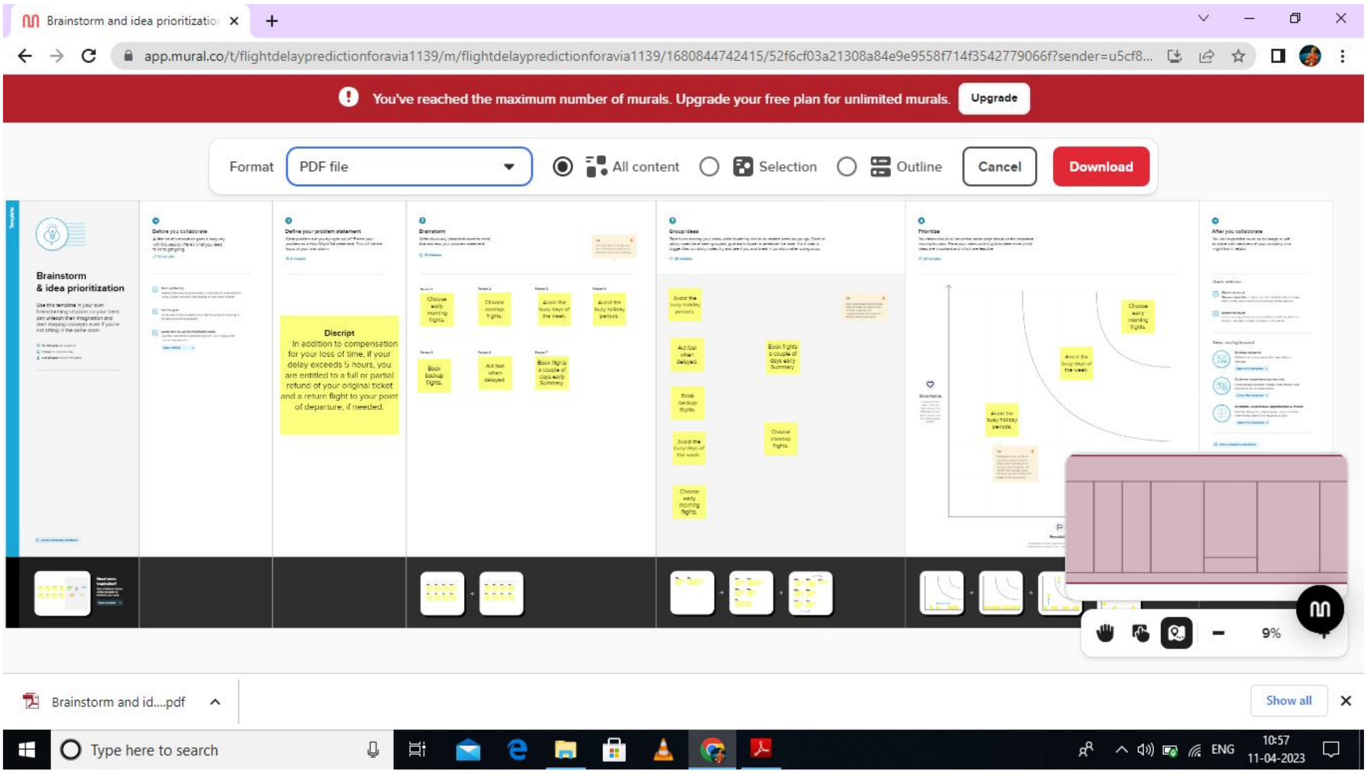
From a business perspective, flight delay prediction can help airlines and airports improve their operations and reduce costs. By identifying and addressing the factors that contribute to flight delays, airlines and airports can take proactive measures to mitigate the impact of delays. This can lead to improved on-time performance, which can help airlines and airports attract and retain customers and increase revenue. Additionally, flight delay prediction can help airlines and airports optimize their staffing and resource allocation, resulting in cost savings.

2.PROBLEM DEFINITION AND DESIGN THINKING

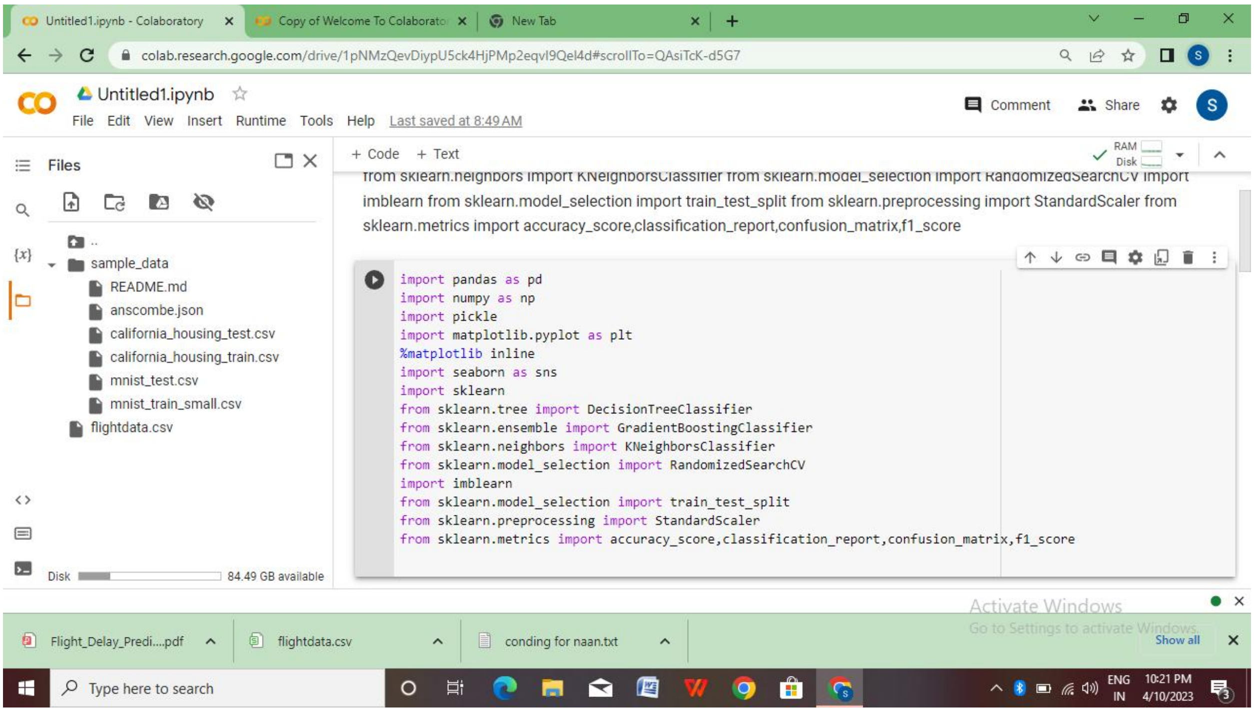
2.1.Empathy map

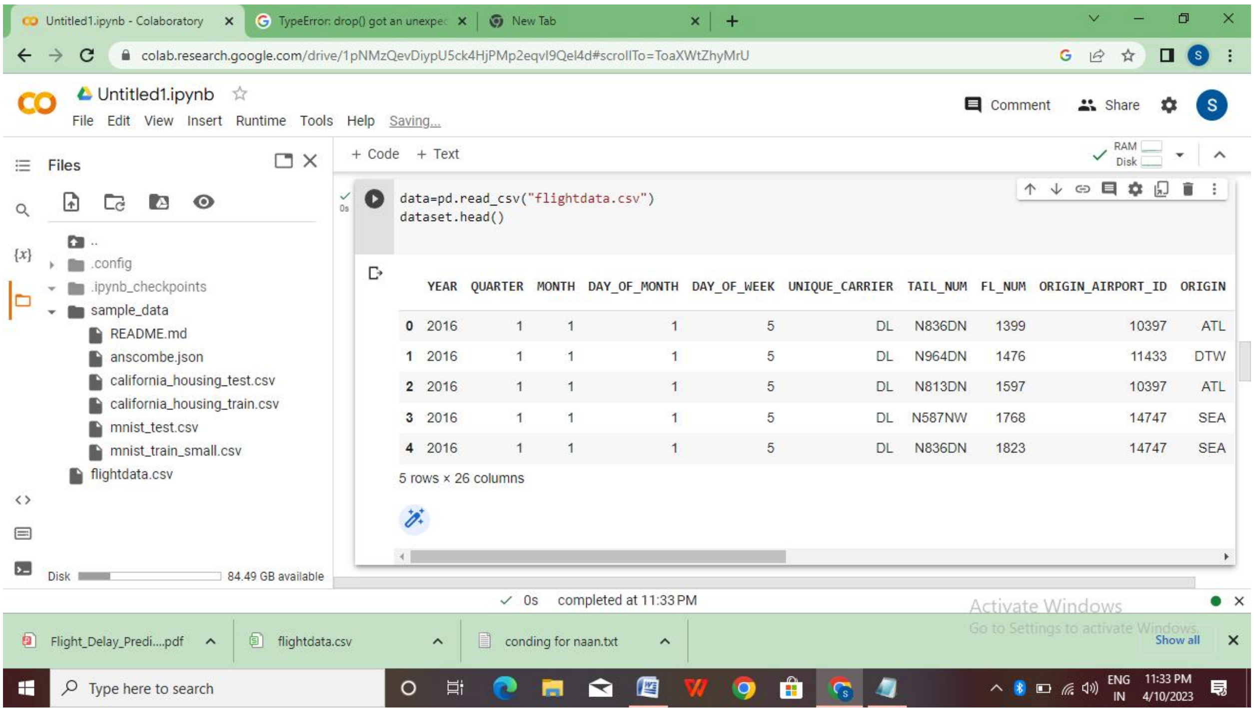


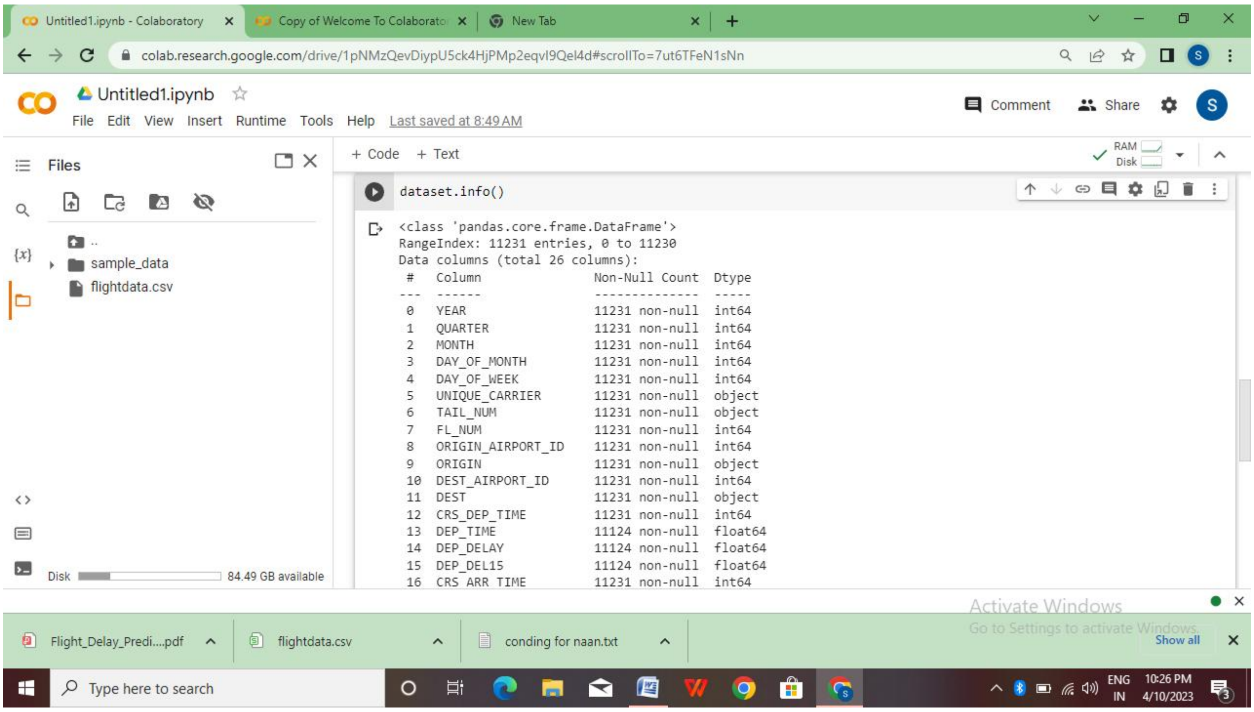
2.2 Ideation and Brainstorming map

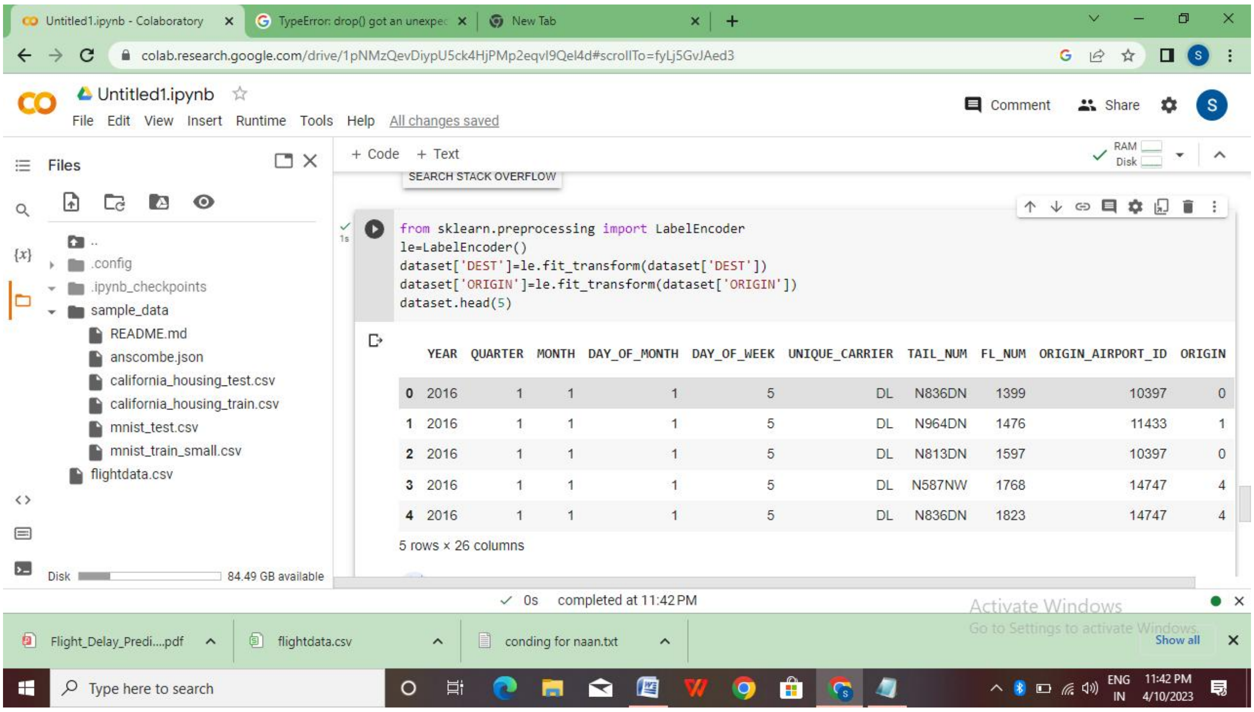


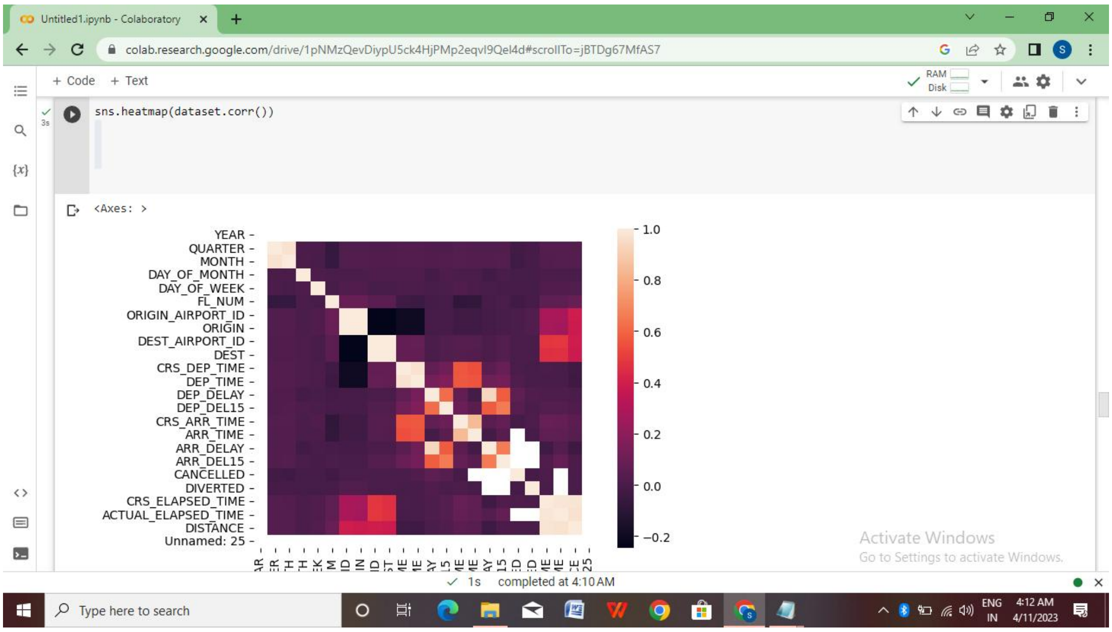
3. RESULTS

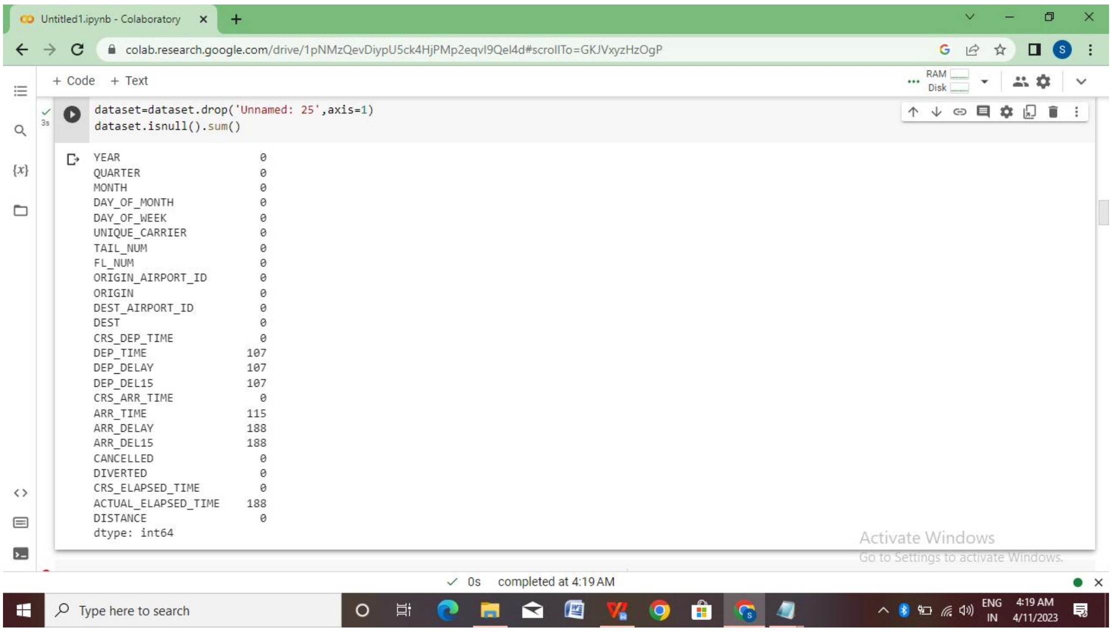

Untitled1.ipynb
File Edit View Insert Runtime Tools Help Last saved at 8:49 AM
Files
sample_data
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
flightdata.csv
from sklearn.neighbors import KNeighborsClassifier from sklearn.model_selection 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
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
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
Disk 84.49 GB available

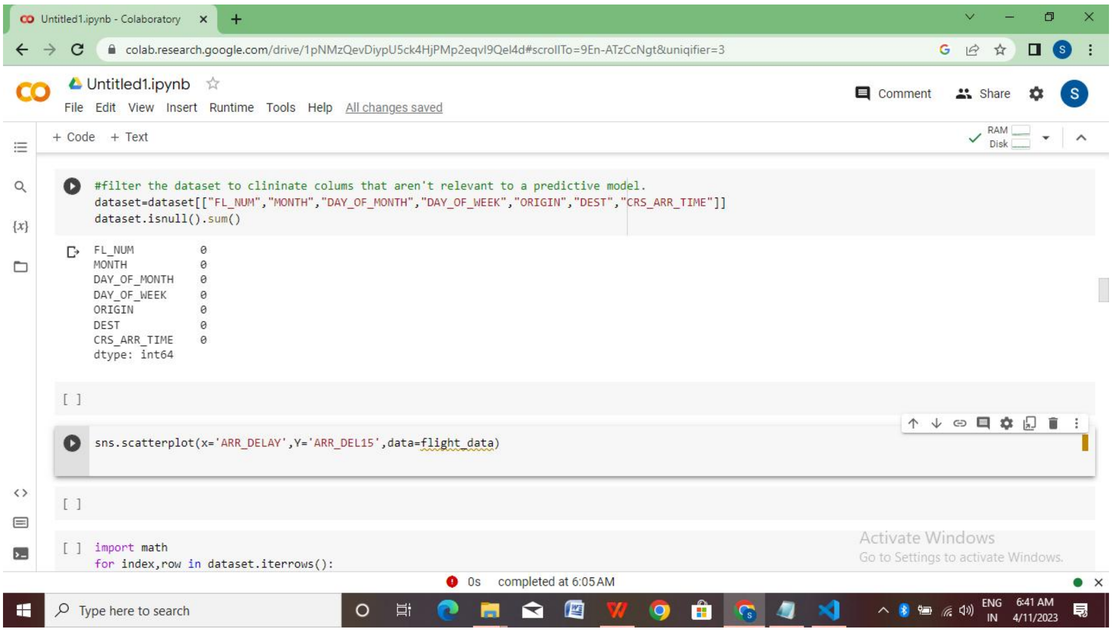

Untitled1.ipynb
File Edit View Insert Runtime Tools Help Saving...
data=pd.read_csv("flightdata.csv")
dataset.head()
YEAR QUARTER MONTH DAY_OF_MONTH DAY_OF_WEEK UNIQUE_CARRIER TAIL_NUM FL_NUM ORIGIN_AIRPORT_ID ORIGIN
0 2016 1 1 1 5 DL N836DN 1399 10397 ATL
1 2016 1 1 1 5 DL N964DN 1476 11433 DTW
2 2016 1 1 1 5 DL N813DN 1597 10397 ATL
3 2016 1 1 1 5 DL N587NW 1768 14747 SEA
4 2016 1 1 1 5 DL N836DN 1823 14747 SEA
5 rows × 26 columns
0s completed at 11:33 PM

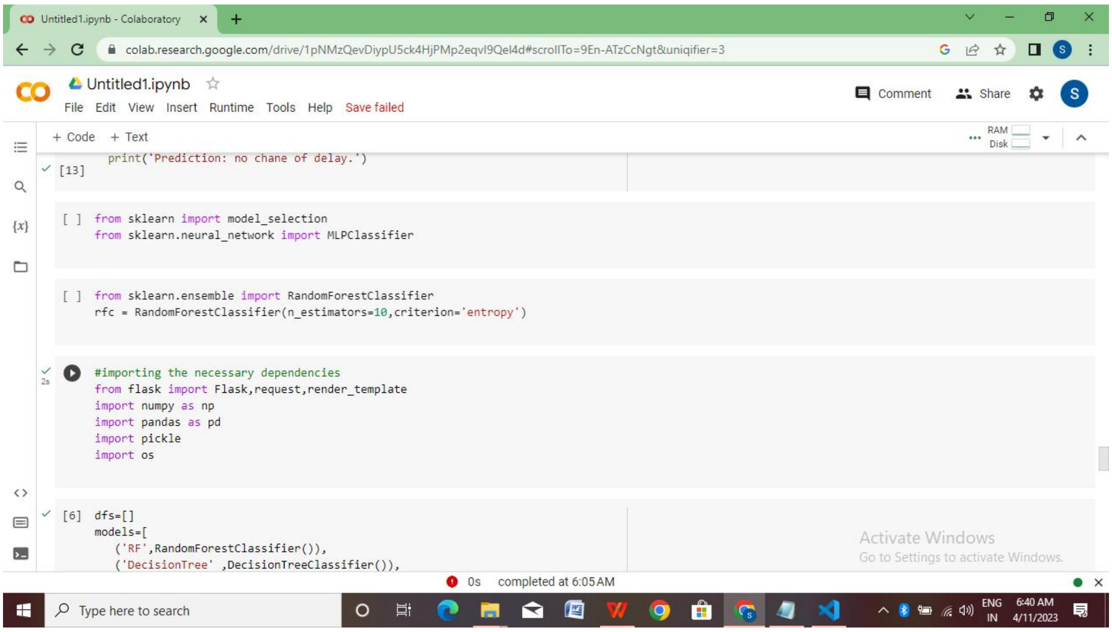

Untitled1.ipynb - Colaboratory
Copy of Welcome To Colaborator
New Tab
colab.research.google.com/drive/1pNMzQevDiypU5ck4HjPMp2eqvl9Qel4d#scrollTo=7ut6TFeN1sNn
Untitled1.ipynb
File Edit View Insert Runtime Tools Help Last saved at 8:49 AM
Comment Share
Files
+ Code + Text
RAM
Disk
..
sample_data
flightdata.csv
Disk 84.49 GB available
dataset.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
Activate Windows
Go to Settings to activate Windows.
Flight_Delay_Predi....pdf
flightdata.csv
conding for naan.txt
Show all
Type here to search
ENG IN
10:26 PM
4/10/2023


Untitled1.ipynb - Colaboratory
TypeError: drop() got an unexpec
New Tab
colab.research.google.com/drive/1pNMzQevDiypU5ck4HjPMp2eqvl9Qel4d#scrollTo=fyLj5GvJAed3
Untitled1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
RAM
Disk
SEARCH STACK OVERFLOW
..
.config
.ipynb_checkpoints
sample_data
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
flightdata.csv
Disk 84.49 GB available
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
dataset['DEST']=le.fit_transform(dataset['DEST'])
dataset['ORIGIN']=le.fit_transform(dataset['ORIGIN'])
dataset.head(5)
YEAR QUARTER MONTH DAY_OF_MONTH DAY_OF_WEEK UNIQUE_CARRIER TAIL_NUM FL_NUM ORIGIN_AIRPORT_ID ORIGIN
0 2016 1 1 1 5 DL N836DN 1399 10397 0
1 2016 1 1 1 5 DL N964DN 1476 11433 1
2 2016 1 1 1 5 DL N813DN 1597 10397 0
3 2016 1 1 1 5 DL N587NW 1768 14747 4
4 2016 1 1 1 5 DL N836DN 1823 14747 4
5 rows × 26 columns
0s completed at 11:42 PM
Activate Windows
Go to Settings to activate Windows.
Flight_Delay_Predi....pdf
flightdata.csv
conding for naan.txt
Show all
Type here to search
ENG IN
11:42 PM
4/10/2023


Untitled1.ipynb - Colaboratory
colab.research.google.com/drive/1pNMzQevDiypU5ck4HjPMp2eqvl9Qel4d#scrollTo=jBTDg67MfAS7
+ Code + Text
sns.heatmap(dataset.corr())
<Axes: >
YEAR
QUARTER
MONTH
DAY_OF_MONTH
DAY_OF_WEEK
FL_NUM
ORIGIN_AIRPORT_ID
ORIGIN
DEST_AIRPORT_ID
DEST
CRS_DEP_TIME
DEP_TIME
DEP_DELAY
DEP_DEL15
CRS_ARR_TIME
ARR_TIME
ARR_DELAY
ARR_DEL15
CANCELLED
DIVERTED
CRS_ELAPSED_TIME
ACTUAL_ELAPSED_TIME
DISTANCE
Unnamed: 25
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
Activate Windows
Go to Settings to activate Windows.
1s completed at 4:10 AM
Type here to search
ENG IN
4:12 AM
4/11/2023


Untitled1.ipynb - Colaboratory
colab.research.google.com/drive/1pNMzQevDiypU5ck4HjPMp2eqvl9Qel4d#scrollTo=GKJVxyzHzOgP
+ Code + Text
dataset=dataset.drop('Unnamed: 25',axis=1)
dataset.isnull().sum()
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
Activate Windows
Go to Settings to activate Windows.
0s completed at 4:19 AM
Type here to search
ENG IN
4:19 AM
4/11/2023


Untitled1.ipynb - Colaboratory
colab.research.google.com/drive/1pNMzQevDiypU5ck4HjPMp2eqvl9Qel4d#scrollTo=9En-ATzCcNgt&uniqifier=3
Untitled1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
#filter the dataset to clininate colums that aren't relevant to a predictive model.
dataset=dataset[["FL_NUM","MONTH","DAY_OF_MONTH","DAY_OF_WEEK","ORIGIN","DEST","CRS_ARR_TIME"]]
dataset.isnull().sum()
FL_NUM 0
MONTH 0
DAY_OF_MONTH 0
DAY_OF_WEEK 0
ORIGIN 0
DEST 0
CRS_ARR_TIME 0
dtype: int64
[ ]
sns.scatterplot(x='ARR_DELAY',Y='ARR_DEL15',data=flight_data)
[ ]
[ ] import math
for index,row in dataset.iterrows():
Activate Windows
Go to Settings to activate Windows.
0s completed at 6:05 AM
Type here to search
ENG IN
6:41 AM 4/11/2023


Untitled1.ipynb - Colaboratory
colab.research.google.com/drive/1pNMzQevDiypU5ck4HjPMp2eqvl9Qel4d#scrollTo=9En-ATzCcNgt&uniqifier=3
Untitled1.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment
Share
+ Code + Text
RAM
Disk
print('Prediction: no chane of delay.')
[13]
[ ] from sklearn import model_selection
from sklearn.neural_network import MLPClassifier
[ ] from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=10,criterion='entropy')
#importing the necessary dependencies
from flask import Flask,request,render_template
import numpy as np
import pandas as pd
import pickle
import os
[6] dfs=[]
models=[
('RF',RandomForestClassifier()),
('DecisionTree' ,DecisionTreeClassifier()),
Activate Windows
Go to Settings to activate Windows.
0s completed at 6:05 AM
Type here to search
ENG IN
6:40 AM 4/11/2023

4.ADVANTAGES AND DISADVANTAGES

4.1ADVANTAGES

- Fast speed
- Rapid service.
- Low infrastructure
- No physical barriers
- Defence service
- Security

4.2 DISADVANTAGES

- Costly service
- Limited capacity
- Undependable and risky
- Accident-prone
- Requires skill
- Unfit for cheap and bulky goods

5.APPLICATIONS

Airport (International & National)

Airline Transport

6.CONCLUSION

The paper performed a prediction of the occurrence of flight delays by adapting it into a machine learning problem. A supervised machine learning approach in the form of binary classification was used for the prediction. Seven algorithms were used for delay prediction, and four measures were used for algorithms performance evaluation. Due to the imbalanced nature of the data set, evaluation measures were weighted to eliminate the dominant effect of non-delayed flights over delayed flights. After applying classifiers to the delay prediction, the values of their four measures were compared to evaluate the performance of each model.

The result shows that the highest values of accuracy, precision, recall, and f1-score are generated by the Decision Tree model (accuracy: 0.9778; precision: 0.9777; recall: 0.9778; f1-score: 0.9778). Such high values indicate that the Decision Tree performs well when predicting flight delays in the data set. Other tree-based ensemble classifiers also show good performance. Random Forest and Gradient Boosted Tree have an accuracy of 0.9240 and 0.9334, significantly higher than the rest of the models. The other four base classifiers Logistic Regression, KNN, Gaussian Naïve Bayes, and SVM, are not tree-based and did not show good performance. The KNN model is the worst performed since its precision and f1-score are the lowest among the seven models.

The data set selected for this paper is imbalanced distributed, which may cause significant variation in the performance of each algorithm. In this paper, this problem was solved by the use of weighted evaluation measures. For future studies, using techniques such as SMOTE can better resolve this imbalance and improve the prediction. The result of algorithm comparison shows that tree-based ensemble algorithms tend to better predict flight delays of this data set. It will be valuable to repeat similar experimental processes using more tree-based ensemble algorithms to discover their significance in flight delay prediction.

7.FUTURE SCOPE

- After studying different models on the dataset, it is observed that KNN provides us the best results with accuracy of about 86%.
- The model accuracy can be increased by taking into the account variables like weather conditions and airline employees efficiency.
- Airlines can determine efficient routes with minimum delay possibility.
- This model can help passengers to plan layover at particular airport.

8.APPENDIX

```
# -*- coding: utf-8 -*-
"""Untitled1.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1pNMzQevDiypU5ck4HjPMp2eqvI9Qel4d

import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
import scaborn as sns
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
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
"""

# Commented out IPython magic to ensure Python compatibility.
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
```

```
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("flightdata.csv")

"""dataset.info()


# New Section
"""

dataset.info()

data=pd.read_csv("flightdata.csv")
dataset.head()

dataset=dataset.drop('Unnamed: 25',axis=1)
dataset.isnull().sum()

#filter the dataset to clininate colums that aren't relevant to a predictive model.
dataset=dataset[["FL_NUM","MONTH","DAY_OF_MONTH","DAY_OF_WEEK","ORIGIN","DEST","CRS_ARR_
TIME"]]
dataset.isnull().sum()



sns.scatterplot(x='ARR_DELAY',Y='ARR_DEL15',data=flight_data)




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'])
odataset.head(5)

dataset['ORIGIN'].unique()
array([0,1,4,3,2])
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

flight_data.describe()

sns.distplot(flight_data.MONTH)

## Decision tree
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)

## RandomForest
y_

from seaborn.axisgrid import FacetGrid
sns.catplot(x"ARR_del15",y="ARR_DELAY",kind='bar',data=flight_data)
<seaborn.axisgrid.FacetGrid at 0x22716099eb0>

sns.heatmap(dataset.corr())

# Testing the model
y_pred = classification.predict(x_test)



from sklearn.preprocessing import standardScaler
sc= standardScaler()
x_train= sc.fit_transform(x_train)
x_test = sc.transform(x_test)

from IPython.utils.text import columnize
from sklearn.metrics.pairwise import DataConversionWarning
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=10,criterion='entropy')
rfc.fit(x_train,y_train)
<ipython-input-125-b87bb2ba9825>:1:DataConversionWarning:A column-vector y warning
ravel().
rfc.fit(x_train,y_train)
RandomForestClassifier(criterion='entropy',n_estimates=10)
y_predict = rfc.predict(x_test)

#creating NN skleton view
classification = sequential()
classification.add(Dense(30,activation='relu'))
classification.add(Dense(128,activation='relu'))
classification.add(Dense(64,activation='relu'))
classification.add(Dense(22,activation='relu'))
classification.add(Dense(1,activation='sigmoid'))
```

```
# complling the ANN model
classification.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])

#Trainig the model
classification.fit(x_train,y_train,batch_size=4,validation_split=0.2,epochs=100)

## RandomForest
y_pred = rfc.predict([[129,99,1,0,0,1,1,1,0,1,1,1,1,1]])
print(y_pred)
(y_pred)

def predict_exit(sample_value):
 # covert list to numpy array
 sample_value = np.array(sample_value)
#Reshape because sample_value contains only 1 record
sample_value = sample_value.reshape(1,-1)
#Feature Scaling
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 chane of delay.')

from sklearn import model_selection
from sklearn.neural_network import MLPClassifier

from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=10,criterion='entropy')

#importing the necessary dependencies
from flask import Flask,request,render_template
import numpy as np
import pandas as pd
import pickle
import os

dfs=[]
models=[
   ('RF',RandomForestClassifier()),
   ('DecisionTree' ,DecisionTreeClassifier()),
   ('ANN',MLPClassifier())
]
results = []
 names = []
 scoring = ['accuracy','precision_weighted', 'recall_weghted', 'f1_weighted', 'roc_auc']
 target_names = ['no delay', 'delay']
 for name,model in models:
      kflod= model_selection.kfold(n_splits=5, shuffle=True, radom_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.appnd(cv_results)
      names.append(name)
      this_df=pd.DataFrame(cv_results)
      this_df['model']=name
      dfs.append(this_df)
final=pd.oncat(dfs, ignore_indext=true)
return final

sns.catplot(x="ARR_DEL15",y="ARR_DELAY",kind='bar',data=flight_data)

model = pickle.load(open('flight.pkl' , 'rb'))
app = Flask(_name_) #initializing the app

# giving some parameters that can be use in randized search cv
parameters={
            'n_estimators' : [1,2,30,55,68,74,90,120,115],
            'criterion' : ['gini' , 'entropy'],
            'max_features' : ["auto" , "sqrt", "log2"],
      'max_depth' : [2,5,8,10] 'verbose' :[1,2,3,4,6,,8,9,10]
}

#performing the randomized cv
RCV = RandomizedSearchCV(estimator=rf,param_distributions=parameters,cv=10,n_iter=4)

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(origin == "jfk"):
```

```
 destination1,destination2,destination3,destination4,destination5  = 0,0,1,0,0
if(origin == "sea"):
  destination1,destination2,destination3,destination4,destination5  = 0,1,0,0,0
if(origin == "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,daofmonth,dayofweek,origin,origin2,origin3,origin4,origin5,destination1,de
stination2,destination3,destination4,destination5]]
 #print(total)
 y_pred = model.predict(total)
print(y_pred)
if(y_pred==[0.]):
  ans="The Flight will be on time"
else:
   ans="The Filght will be delayed"
return render_template("index.html",showcase = ans)

if _name_=='_main_':
  app.run(debug = True)
```