GOVERNMENT OF TAMILNADU

Naan Muthalvan - Project-Based Experiential Learning

Flight Delay Prediction for aviation Industry using Machine Learning

Submitted by

Team ID NM2023TMID20562

R.KEERTHANA - (20326ER012)
J.JOVIYA - (20326ER009)
V.KARTHIKA - (20326ER010)
R. KAVERI KAVITHA - (20326ER011)

Under the guidance of

Mrs. J. SUKANYA, MCA., M.Phil.,
Assistant Professor

PG and Research Department of Computer Science

M.V.MUTHIAH GOVERNMENT ARTS COLLEGE FOR WOMEN
(Affiliated To Mother Teresa Women' s University, Kodaikanal)
Reaccredited with " A" Grade by NAAC
DINDIGUL-624001.

APRIL - 2023

M.V.MUTHIAH GOVERNMENT ARTS COLLEG FOR WOMEN
(Affiliated to Mother Teresa Women' s University, Kodaikanal)
Reaccredited with " A" Grade by NAAC Dindigul - 624 001

PG & RESEARCH DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this is a bonafide record of the project entitled, **" FLIGHT DELAY PREDICTION FOR AVIATION INDUSTRY USING MACHINE LEARNING** done **by Ms.J. JOVIYA (20326ER009), Ms.V. KARTHIKA (20326ER010), Ms.R. KAVERI KAVITHA (20326ER011), and Ms.R. KEERTHANA (20326ER012)**. This is submitted in partial fulfillment for the award of the degree of **Bachelor of Science in Computer Science in M.V.MUTHIAH GOVERNMENT ARTS COLLEGE FOR WOMEN, DINDIGUL** during the period of December 2022 to April 2023.

Project Mentor(s) Head of the Department

Submitted for viva-voce Examination held on _____11.04.2023_____

TABLE OF CONTENTS

ABSTRACT

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, 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. INTRODUCTION

1.1 OVERVIEW

Flight delays can cause significant disruptions to travel plans and can have a negative impact on airline operations. As a result, predicting flight delays has become a critical issue for airlines and airports. In this project, we aim to build a machine learning model that can predict the likelihood of flight delays based on historical data. The model will be trained on a dataset containing information such as flight schedules, weather conditions, and other factors that can influence flight delays. To achieve this, we will use various machine learning algorithms such as decision trees, random forests, and neural networks. We will also perform feature engineering to extract useful information from the dataset Once the model is trained and evaluated, it can be used to predict the likelihood of flight delays for future flights, allowing airlines and airports to better plan for potential disruptions. Overall, this project aims to develop a useful tool for the aviation industry that can improve the accuracy of flight delay predictions, ultimately benefiting both airlines and traveller alike.

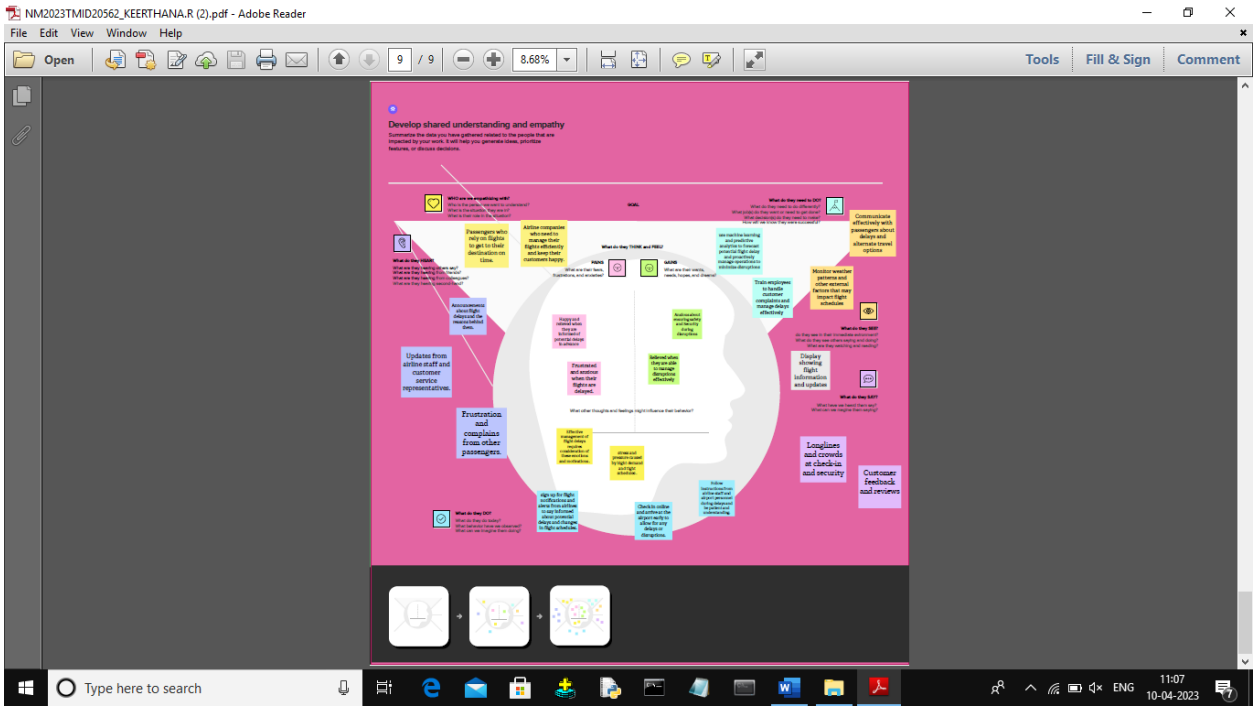
1.2 PURPOSE

There are plenty of other reasons why passengers face flight,delay, such as time for fueling, boarding passengers, aircraft cleaning, etc. Airline allow for a little bit of flexibility,and it's important for passengers to understand such so they can fit some flexibility into their schedule,too.

2.PROBLEM DEFINITION

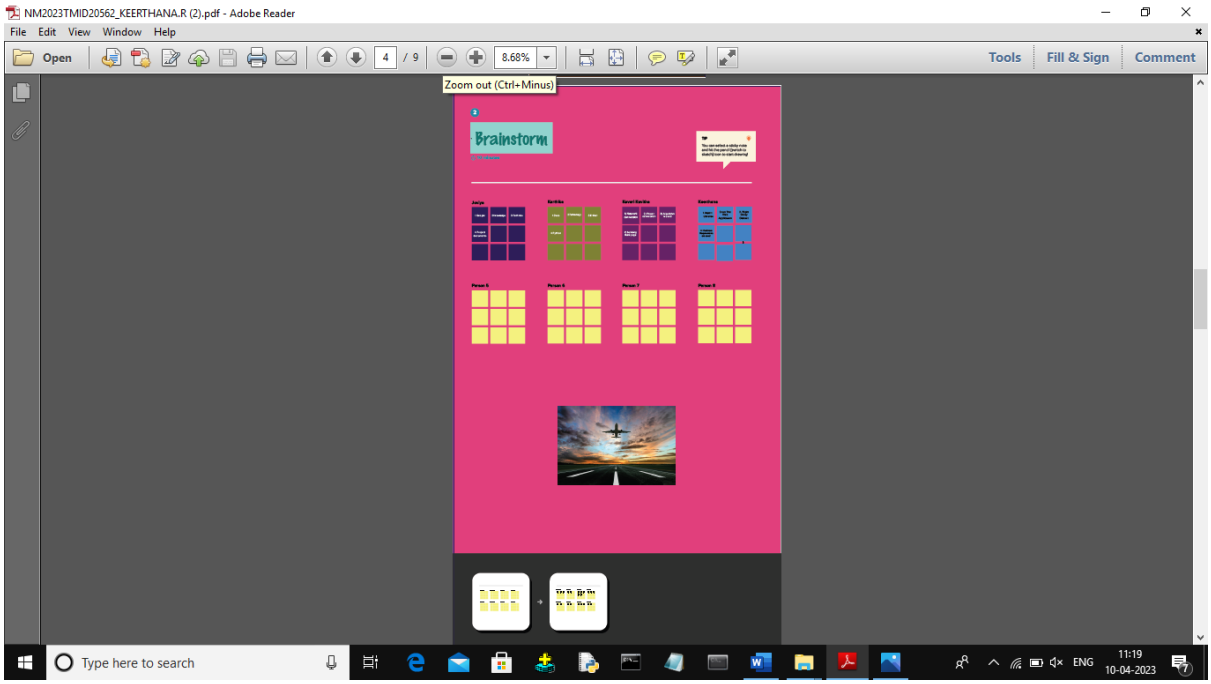
2.1 Empathy Map

Using this empathy mapcanvas, , you can identify the needs and concerns of your passenger, which can help inform your approach to predicting and managing flight delays. For exzmple: Your may want to provide real-time update on the flight status or offer alternative transportation options to easy their anxiety and help them plan ahead.



2.2 Ideation & Brainstroming Map

Brainstroming is a group problem -solving method that involves the spontaneous contribution of ccreative ideas and solutions. This technique requiresss intensive , freewheeling discussion in which every member of the group is encouraged to think aloud and suggest as many ideas as possible based on their diverse knowledge.



3.RESULT

WEB BROWSER AND WRITE THE LOCALHOST URL:


localhost:5000/prediction
Prediction of Flight Delay
Enter the Flight Number :
Month :
Day of Month :
Day of Week :
origin MSP
destination MSP
Scheduled Departure Time :
Scheduled Arrival Time :
Actual Departure Time :
SUBMIT
The Flight will be on time

4. ADVANTAGES & DISADVANTAGES

4.1 Advantages of flight delay prediction project:

1. Safety: Knowing in advance that a flight may be delayed due to adverse weather conditions or other factors can help airlines make better decisions about whether to cancel or delay a flight to ensure passenger safety.
2. Improved customer satisfaction: With accurate predictions of flight delays, airlines can proactively notify passengers and offer alternative travel arrangements, which can help prevent frustration and inconvenience for travelers.
3. Cost savings: Predicting flight delays can help airlines avoid unnecessary expenses, such as overtime pay for crew members, additional fuel costs, and expenses associated with rerouting passengers.
4. Operational efficiency: By predicting flight delays, airlines can adjust their operations, such as scheduling crew members and equipment, to minimize the impact of delays on their overall performance.
5. Competitive advantage: By offering more reliable and predictable travel options, airlines can gain a competitive advantage over other carriers in the market.

4.2 Disadvantages of flight delay prediction project:

1. Implementation costs: Developing and implementing a flight delay prediction project can be expensive and require significant investment in technology and personnel. There may also be ongoing costs associated with maintaining and updating the system.

2. Resource allocation: Prediction flight delay can help airlines optimize their resources. However, allocating resources based on predictions that the may not be accurate can lead to wasted resources and increased costs.

5.APPLICATIONS

1. Collect data on flight delays: There are several sources of data on flight delays, including public datasets, APIs, and scraping flight information from airline websites. You'll need to determine which data source(s) are most appropriate for your project.

2. Explore the data: Once the data is cleaned and preprocessed, you can start to explore it to identify patterns and trends. You may want to use descriptive statistics or visualizations to better understand the data.

3. Develop a model: Depending on the goals of your project, you may want to develop a predictive model to forecast flight delays. You'll need to select appropriate features and choose a modeling approach that works well with your data.

4. Evaluate the model: Once you've developed a model, you'll want to evaluate its performance. You may want to use metrics such as accuracy, precision, or recall to assess how well the model is predicting flight delays.

5. Deploy the application: Finally, you'll want to deploy your application so that users can interact with it. You may want to build a web application, a mobile app, or integrate it with an existing platform.

6. CONCLUSION

In conclusion, the flight delay prediction project aims to build a machine learning model that can accurately predict the likelihood of flight delays based on historical flight data. The project involves various steps such as data pre processing, feature engineering, model selection, and evaluation. By predicting the likelihood of flight delays, the model can be used by airlines and airports to better plan and manage their operations. This can help airlines adjust their schedules in advance, minimize the impact of delays, and improve the travel experience for passengers. The project has used various machine learning algorithms such as decision trees, random forests, and neural networks, along with feature engineering and data pre processing techniques. The performance of the model has been evaluated using various metrics, and the best performing model can be deployed for real-time prediction of flight delays. Overall, the project has the potential to make a significant impact on the aviation industry, improving airline operations, reducing passenger frustration, and enhancing the overall travel experience.

7.FUTURE SCOPE

There are several possible future enhancements that can be considered for the Flight Delay Prediction project, including:

Using ensemble learning: Ensemble learning is a technique where multiple models are combined to produce a more accurate prediction. Implementing ensemble learning techniques such as stacking or bagging can help improve the overall accuracy of the model. While the project already includes several factors that can affect flight delays, other factors such as the airline's safety record, the aircraft's maintenance history, and flight crew availability can also be considered to improve the accuracy of the model. Feature engineering plays a crucial role in building accurate machine learning models Overall, the Flight Delay Prediction project offers several opportunities for future enhancements that can improve the accuracy and usability of the model in real-world scenarios.

8. APPENDIX

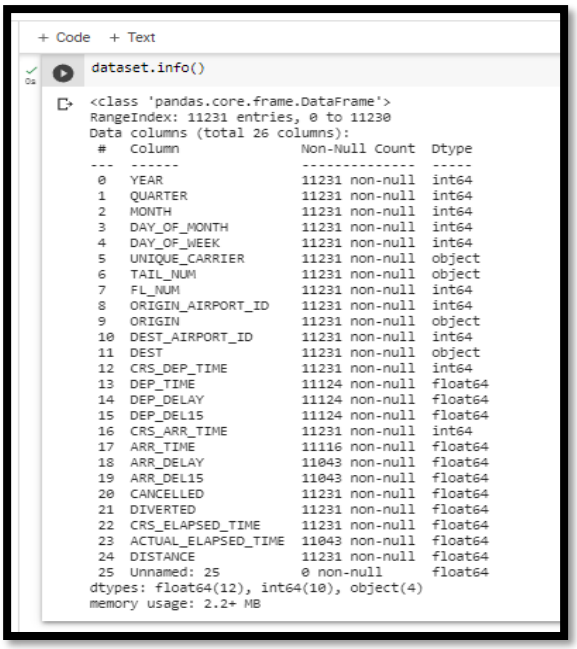
A. SOURCE CODE

```
+ Code  + Text

import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plot
%matplotlib inline
import seaborn as sns
import sklearn
import pandas
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
```

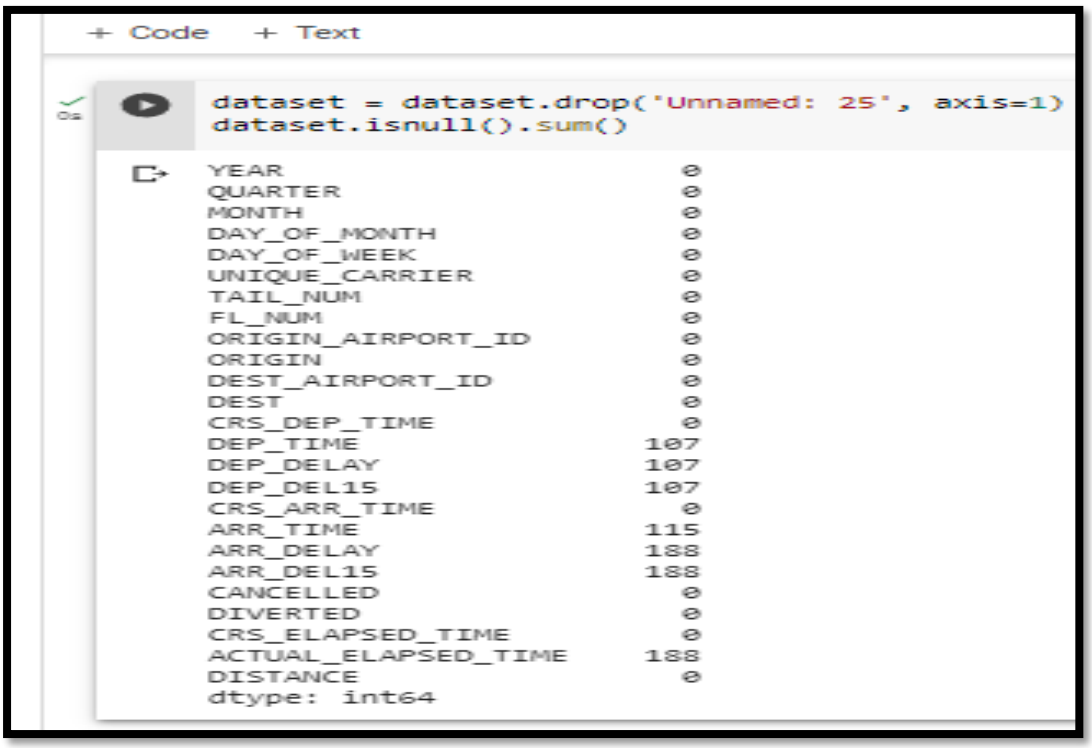
```
dataset=pd.read_csv("/content/flightdata.csv")
dataset.head()
```

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	TAIL_NUM	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	...	CRS_AR
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	...	

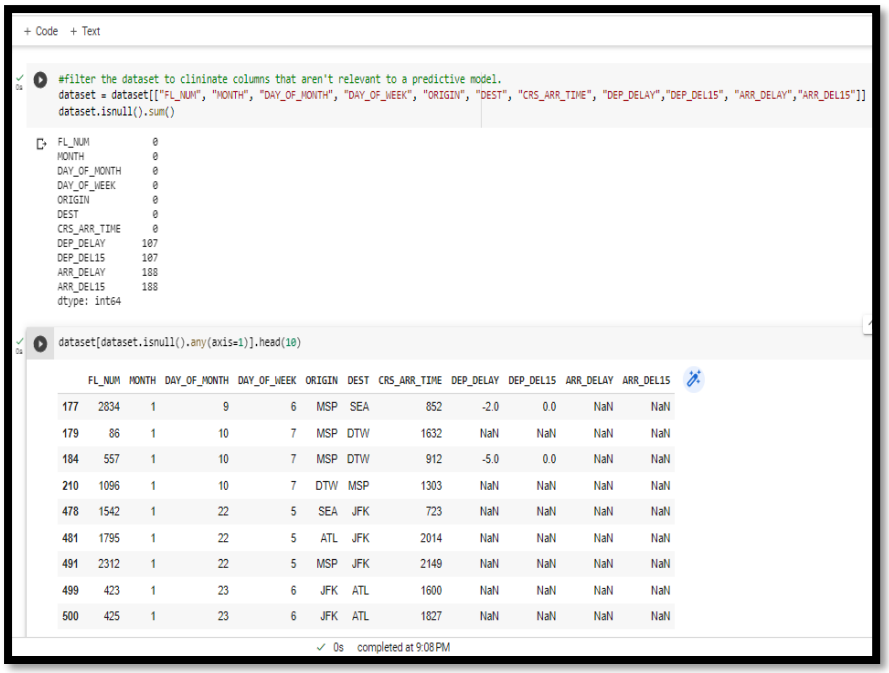
5 rows × 26 columns



```
+ Code  + Text

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
 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
```



```
+ 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
```



```
+ Code + Text

#filter the dataset to cliniate columns that aren't relevant to a predictive model.
dataset = dataset[["FL_NUM", "MONTH", "DAY_OF_MONTH", "DAY_OF_WEEK", "ORIGIN", "DEST", "CRS_ARR_TIME", "DEP_DELAY","DEP_DEL15", "ARR_DELAY","ARR_DEL15"]]
dataset.isnull().sum()

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

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

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DELAY	DEP_DEL15	ARR_DELAY	ARR_DEL15
177	2834	1	9	6	MSP	SEA	852	-2.0	0.0	NaN	NaN
179	86	1	10	7	MSP	DTW	1632	NaN	NaN	NaN	NaN
184	557	1	10	7	MSP	DTW	912	-5.0	0.0	NaN	NaN
210	1096	1	10	7	DTW	MSP	1303	NaN	NaN	NaN	NaN
478	1542	1	22	5	SEA	JFK	723	NaN	NaN	NaN	NaN
481	1795	1	22	5	ATL	JFK	2014	NaN	NaN	NaN	NaN
491	2312	1	22	5	MSP	JFK	2149	NaN	NaN	NaN	NaN
499	423	1	23	6	JFK	ATL	1600	NaN	NaN	NaN	NaN
500	425	1	23	6	JFK	ATL	1827	NaN	NaN	NaN	NaN

✓ 0s completed at 9:08 PM

```
+ Code + Text
```

500	425	1	23	6	JFK	ATL	1827	NaN	NaN	NaN	NaN
501	427	1	23	6	JFK	SEA	1053	NaN	NaN	NaN	NaN

```
[11] dataset['DEP_DEL15'].mode()

0    0.0
Name: DEP_DEL15, dtype: float64

[12] dataset=dataset.fillna({'ARR_DEL15': 1})
     dataset=dataset.fillna({'DEP_DEL15': 0})
     dataset=dataset.fillna({'ARR_DELAY': 1})
     dataset=dataset.fillna({'DEP_DELAY': 0})
     dataset.iloc[177:185]
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DELAY	DEP_DEL15	ARR_DELAY	ARR_DEL15
177	2834	1	9	6	MSP	SEA	852	-2.0	0.0	1.0	1.0
178	2839	1	9	6	DTW	JFK	1724	-4.0	0.0	-15.0	0.0
179	86	1	10	7	MSP	DTW	1632	0.0	0.0	1.0	1.0
180	87	1	10	7	DTW	MSP	1649	24.0	1.0	14.0	0.0
181	423	1	10	7	JFK	ATL	1600	11.0	0.0	7.0	0.0
182	440	1	10	7	JFK	ATL	849	-1.0	0.0	-14.0	0.0
183	485	1	10	7	JFK	SEA	1945	65.0	1.0	10.0	0.0
184	557	1	10	7	MSP	DTW	912	-5.0	0.0	1.0	1.0

```
[13] import math

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

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DELAY	DEP_DEL15	ARR_DELAY	ARR_DEL15
0	1399	1	1	5	ATL	SEA	21	2.0	0.0	-41.0	0.0
1	1476	1	1	5	DTW	MSP	14	-1.0	0.0	4.0	0.0
2	1597	1	1	5	ATL	SEA	12	2.0	0.0	-33.0	0.0
3	1768	1	1	5	SEA	MSP	13	1.0	0.0	10.0	0.0
4	1823	1	1	5	SEA	DTW	6	-4.0	0.0	8.0	0.0

```
[14] from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     dataset['DEST'] = le.fit_transform(dataset['DEST'])
     dataset['ORIGIN'] = le.fit_transform(dataset['ORIGIN'])
```

Help All changes saved

+ Code + Text

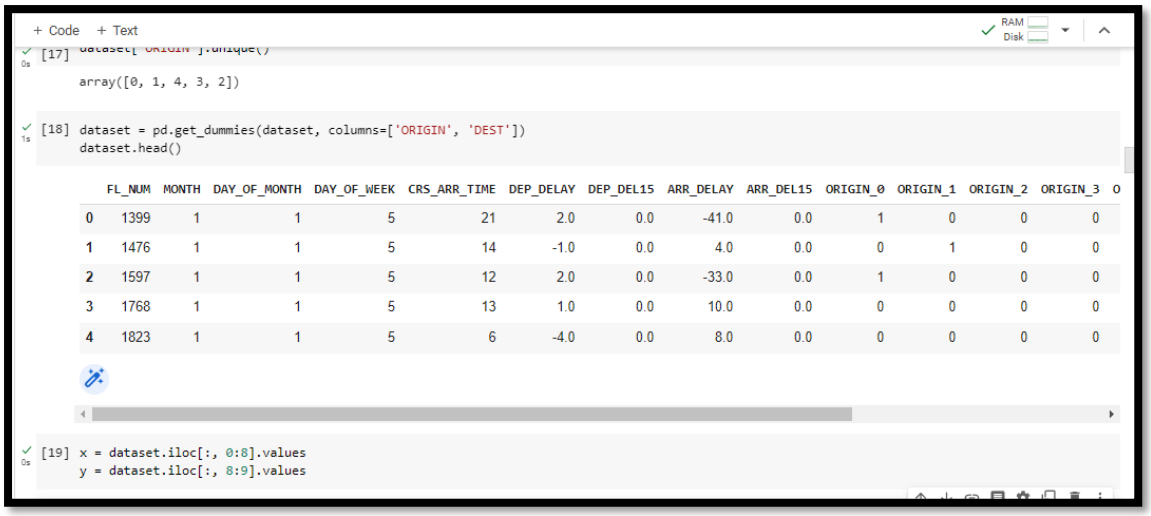
```
[14] dataset['DEST'] = le.fit_transform(dataset['DEST'])
     dataset['ORIGIN'] = le.fit_transform(dataset['ORIGIN'])

[16] dataset.head(5)
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DELAY	DEP_DEL15	ARR_DELAY	ARR_DEL15
0	1399	1	1	5	0	4	21	2.0	0.0	-41.0	0.0
1	1476	1	1	5	1	3	14	-1.0	0.0	4.0	0.0
2	1597	1	1	5	0	4	12	2.0	0.0	-33.0	0.0
3	1768	1	1	5	4	3	13	1.0	0.0	10.0	0.0
4	1823	1	1	5	4	1	6	-4.0	0.0	8.0	0.0

```
dataset['ORIGIN'].unique()

array([0, 1, 4, 3, 2])

[ ] dataset = pd.get_dummies(dataset, columns=['ORIGIN', 'DEST'])
    dataset.head()
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	CRS_ARR_TIME	DEP_DELAY	DEP_DEL15	ARR_DELAY	ARR_DEL15	ORIGIN_0	ORIGIN_1	ORIGIN_2	ORIGIN_3
0	1399	1	1	5	21	2.0	0.0	-41.0	0.0	1	0	0	0
1	1476	1	1	5	14	-1.0	0.0	4.0	0.0	0	1	0	0

0s completed at 9:24PM



+ Code + Text

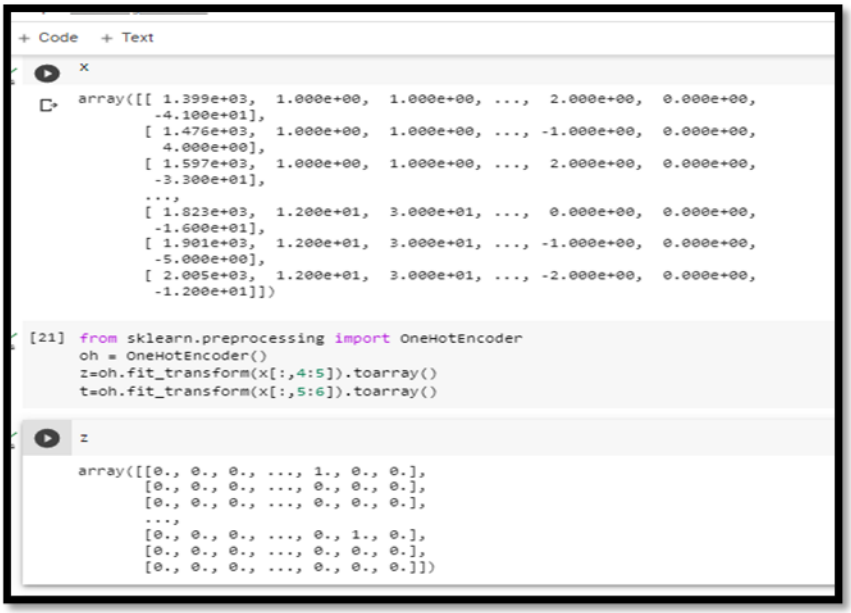
[17] dataset['ORIGIN'].unique()

```
array([0, 1, 4, 3, 2])
```

[18]
```
dataset = pd.get_dummies(dataset, columns=['ORIGIN', 'DEST'])
dataset.head()
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	CRS_ARR_TIME	DEP_DELAY	DEP_DEL15	ARR_DELAY	ARR_DEL15	ORIGIN_0	ORIGIN_1	ORIGIN_2	ORIGIN_3	O
0	1399	1	1	5	21	2.0	0.0	-41.0	0.0	1	0	0	0	
1	1476	1	1	5	14	-1.0	0.0	4.0	0.0	0	1	0	0	
2	1597	1	1	5	12	2.0	0.0	-33.0	0.0	1	0	0	0	
3	1768	1	1	5	13	1.0	0.0	10.0	0.0	0	0	0	0	
4	1823	1	1	5	6	-4.0	0.0	8.0	0.0	0	0	0	0	

[19]
```
x = dataset.iloc[:, 0:8].values
y = dataset.iloc[:, 8:9].values
```



+ Code + Text

```
x
```

```
array([[ 1.399e+03,  1.000e+00,  1.000e+00, ...,  2.000e+00,  0.000e+00,
        -4.100e+01],
       [ 1.476e+03,  1.000e+00,  1.000e+00, ..., -1.000e+00,  0.000e+00,
         4.000e+00],
       [ 1.597e+03,  1.000e+00,  1.000e+00, ...,  2.000e+00,  0.000e+00,
        -3.300e+01],
       ...,
       [ 1.823e+03,  1.200e+01,  3.000e+01, ...,  0.000e+00,  0.000e+00,
        -1.600e+01],
       [ 1.901e+03,  1.200e+01,  3.000e+01, ..., -1.000e+00,  0.000e+00,
        -5.000e+00],
       [ 2.005e+03,  1.200e+01,  3.000e+01, ..., -2.000e+00,  0.000e+00,
        -1.200e+01]])
```

[21]
```
from sklearn.preprocessing import OneHotEncoder
oh = OneHotEncoder()
z=oh.fit_transform(x[:,4:5]).toarray()
t=oh.fit_transform(x[:,5:6]).toarray()
```

```
z
```

```
array([[0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

Help

+ Code + Text

RAM Disk

```
t
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

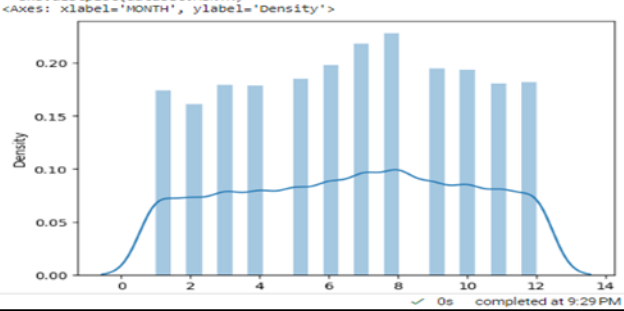
```
[24] x=np.delete(x,[4,5],axis=1)
```

```
dataset.describe()
```

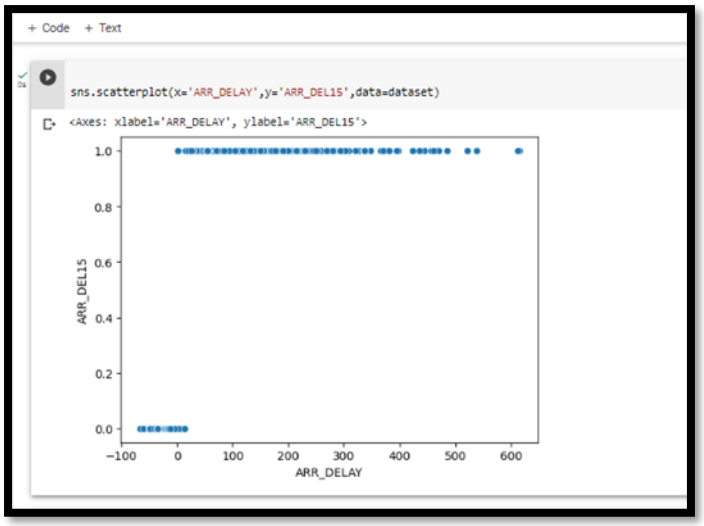
	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	CRS_ARR_TIME	DEP_DELAY	DEP_DEL15	ARR_DELAY	ARR_DEL15	ORIGIN_0	ORIGIN_1	ORIGIN_2
count	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000
mean	1334.325617	6.628973	15.790758	3.960199	15.067314	8.379663	0.141483	-2.513311	0.139168	0.276022	0.195975	0.122340
std	811.875227	3.354678	8.782056	1.995257	5.023534	36.596640	0.348535	38.905443	0.346138	0.447048	0.396967	0.327693
min	7.000000	1.000000	1.000000	1.000000	0.000000	-16.000000	0.000000	-67.000000	0.000000	0.000000	0.000000	0.000000
25%	624.000000	4.000000	8.000000	2.000000	11.000000	-3.000000	0.000000	-19.000000	0.000000	0.000000	0.000000	0.000000
50%	1267.000000	7.000000	16.000000	4.000000	15.000000	-1.000000	0.000000	-10.000000	0.000000	0.000000	0.000000	0.000000
75%	2032.000000	9.000000	23.000000	6.000000	19.000000	4.000000	0.000000	1.000000	0.000000	1.000000	0.000000	0.000000
max	2853.000000	12.000000	31.000000	7.000000	23.000000	645.000000	1.000000	615.000000	1.000000	1.000000	1.000000	1.000000

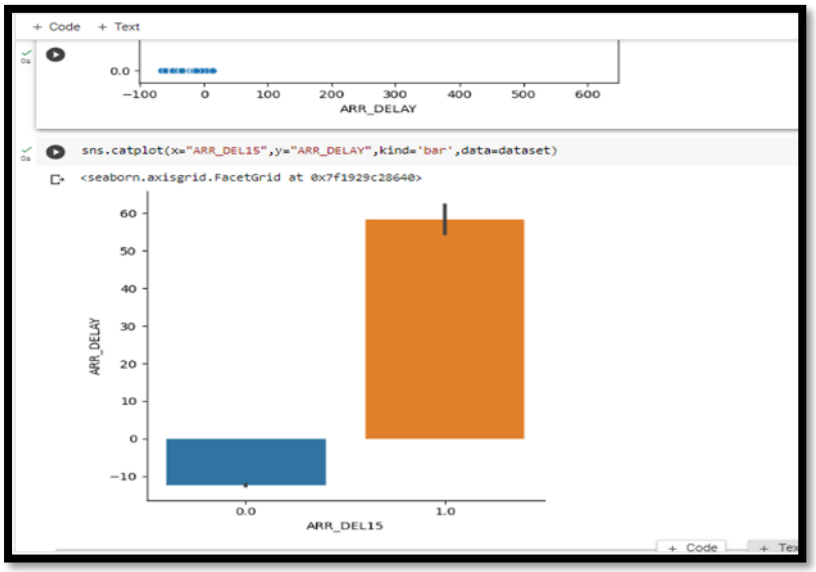
```
sns.distplot(dataset.MONTH)
```

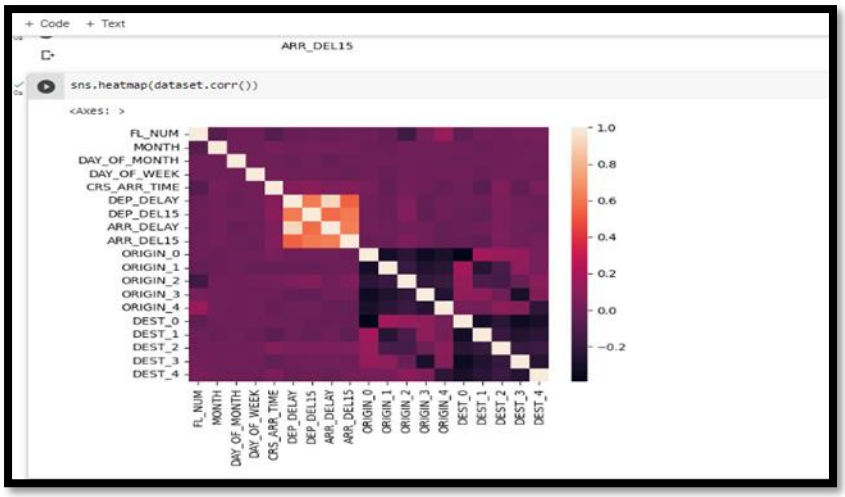
```
<ipython-input-26-43f5c122a6ef>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataset.MONTH)
<Axes: xlabel='MONTH', ylabel='Density'>
```



✓ 0s completed at 9:29 PM


+ Code + Text
sns.scatterplot(x='ARR_DELAY',y='ARR_DEL15',data=dataset)
<Axes: xlabel='ARR_DELAY', ylabel='ARR_DEL15'>
ARR_DEL15
ARR_DELAY

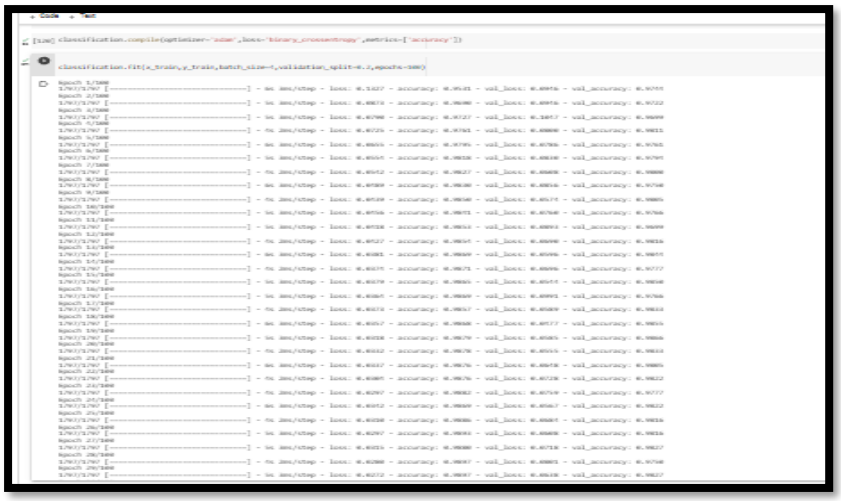

+ Code + Text
sns.catplot(x="ARR_DEL15",y="ARR_DELAY",kind='bar',data=dataset)
<seaborn.axisgrid.FacetGrid at 0x7f1929c28640>
ARR_DELAY
ARR_DEL15


+ Code
+ Text
ARR_DEL15
sns.heatmap(dataset.corr())
<Axes: >
FL_NUM
MONTH
DAY_OF_MONTH
DAY_OF_WEEK
CRS_ARR_TIME
DEP_DELAY
DEP_DEL15
ARR_DELAY
ARR_DEL15
ORIGIN_0
ORIGIN_1
ORIGIN_2
ORIGIN_3
ORIGIN_4
DEST_0
DEST_1
DEST_2
DEST_3
DEST_4
1.0
0.8
0.6
0.4
0.2
0.0
−0.2


+ Code
+ Text
[30] 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)
[31] 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)
[32] x_test.shape
(2247, 6)
x_train.shape
(8984, 6)
[34] y_test.shape
(2247, 1)
[35] y_train.shape
(8984, 1)
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)

```
+ Code + Text

from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(random_state=0)
classifier.fit(x_train, y_train)

DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)

[38] decisiontree = classifier.predict(x_test)

[39] decisiontree

array([1., 0., 0., ..., 0., 0., 1.])

[40] from sklearn.metrics import accuracy_score
desacc = accuracy_score(y_test,decisiontree)

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

rfc.fit(x_train,y_train)

<ipython-input-42-b87bb2ba9825>:1: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for
  rfc.fit(x_train,y_train)
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

```
+ Code + Text

<ipython-input-116-b87bb2ba9825>:1: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape
  rfc.fit(x_train,y_train)
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=10)

[117] y_predict = rfc.predict(x_test)

[118] 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'))

[120] classification.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```



ols Help Save failed

+ Code + Text

```
y_pred = classifier.predict([[129,99,1,0,0,0]])
print(y_pred)
(y_pred)
```

```
[0.]
array([0.])
```

```
[123] y_pred=rfc.predict([[129,99,1,0,0,0]])
      print(y_pred)
      (y_pred)
```

```
[0.]
array([0.])
```

```
[124] classification.save('flight.h5')
```

```
[125] y_pred=classification.predict(x_test)
```

```
71/71 [==============================] - 0s 2ms/step
```

```
[126] y_pred
```

```
array([[1.0000000e+00],
       [0.0000000e+00],
       [1.6780123e-20],
       ...,
       [3.2196211e-19],
       [1.4602942e-25],
       [1.0000000e+00]], dtype=float32)
```

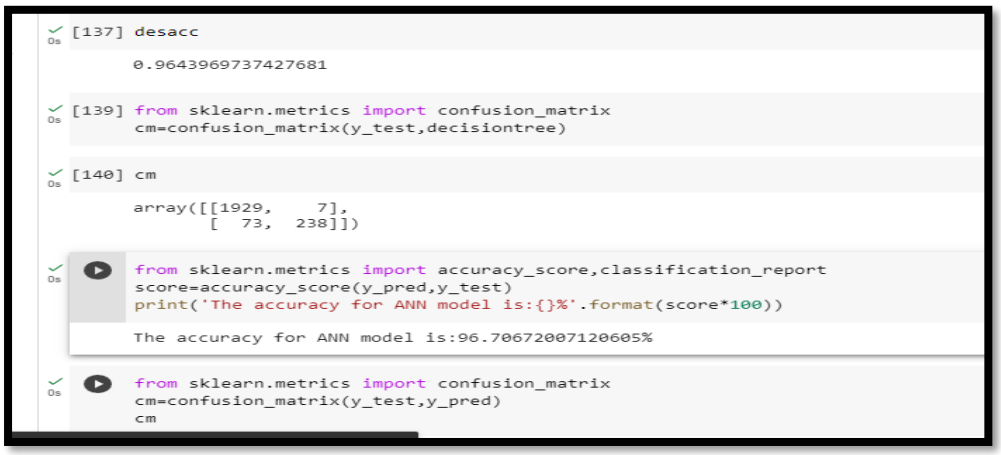
```
+ Code  + Text

y_pred=(y_pred>0.5)
y_pred

array([[ True],
       [False],
       [False],
       ...,
       [False],
       [False],
       [ True]])

+ Code  + Text

[128] 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)

[129] 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')

      1/1 [==============================] - 0s 40ms/step
      Prediction: No chance of delay

[130] from sklearn import model_selection
      from sklearn.neural_network import MLPClassifier
```

```
+ Code  + Text
[130] 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)

/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,
  estimator.fit(X_train, y_train, **fit_params)
<ipython-input-131-4982c366b9c7>:14: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  clf=model.fit(x_train,y_train)
RF
              precision    recall  f1-score   support

    no delay       0.96      1.00      0.98      1936
       delay       0.99      0.77      0.86       311

    accuracy                           0.97      2247
   macro avg       0.98      0.88      0.92      2247
weighted avg       0.97      0.97      0.96      2247

DecisionTree
              precision    recall  f1-score   support
```

ols Help Save failed

+ Code + Text

```
[213] y_pred = y_pred.ravel()

from sklearn.metrics import accuracy_score
print ('Testing accuracy:', accuracy_score(y_test,y_predict))

Testing accuracy: 0.9666221628838452

[133] from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_predict)
cm

array([[1934,    2],
       [  73,  238]])

from sklearn.metrics import accuracy_score
desacc= (y_test,decisiontree)

[137] desacc

0.9643969737427681
```



```
[137] desacc

0.9643969737427681

[139] from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,decisiontree)

[140] cm

array([[1929,    7],
       [  73,  238]])

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))

The accuracy for ANN model is:96.70672007120605%

from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
cm
```


Tools Help Save failed
+ Code + Text
[143] 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]
}
[144] RCV = RandomizedSearchCV(estimator=rfc, param_distributions=parameters, cv=10, n_iter=4)
[145] RCV.fit(x_train,y_train)
[Parallel(n_jobs=1)]: Done 3 out of 3 | elapsed: 0.0s remaining: 0.0s
[Parallel(n_jobs=1)]: Done 4 out of 4 | elapsed: 0.0s remaining: 0.0s
[Parallel(n_jobs=1)]: Done 5 out of 5 | elapsed: 0.0s remaining: 0.0s
[Parallel(n_jobs=1)]: Done 6 out of 6 | elapsed: 0.0s remaining: 0.0s
[Parallel(n_jobs=1)]: Done 7 out of 7 | elapsed: 0.0s remaining: 0.0s
[Parallel(n_jobs=1)]: Done 8 out of 8 | elapsed: 0.0s remaining: 0.0s
[Parallel(n_jobs=1)]: Done 9 out of 9 | elapsed: 0.0s remaining: 0.0s

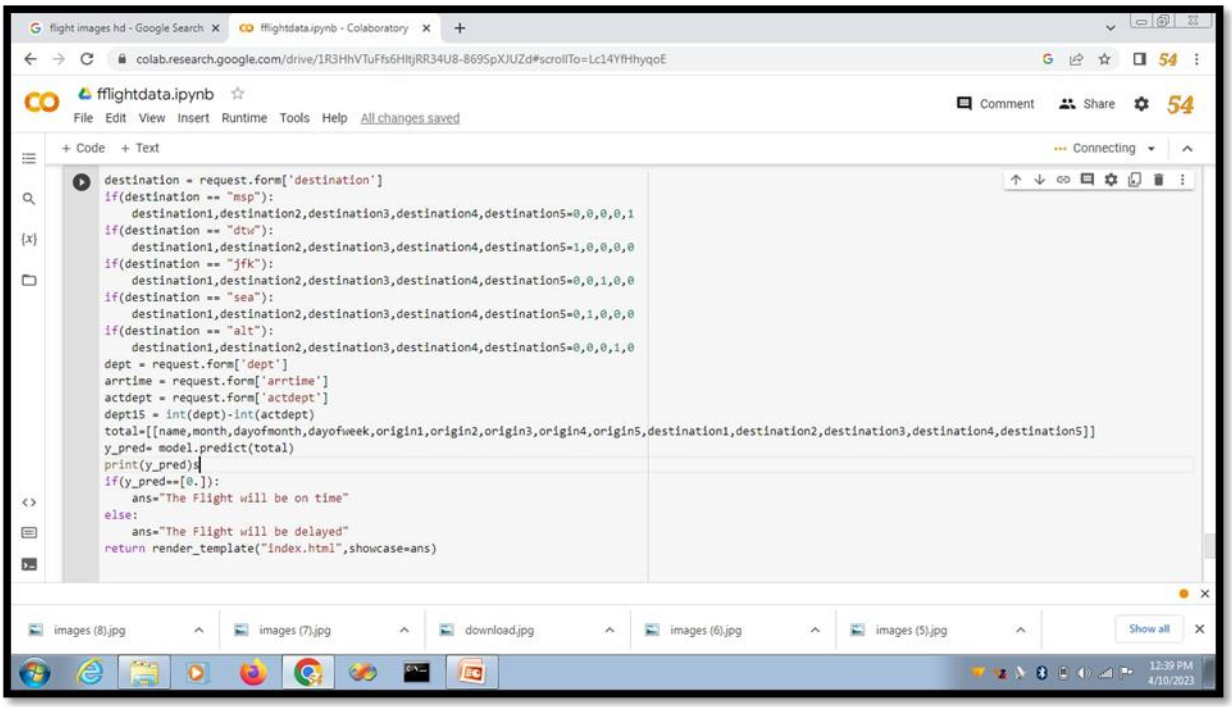

ools Help Save failed
Comment Share
+ Code + Text
RAM Disk
[146] bt_params=RCV.best_params_
bt_score=RCV.best_score_
[147] bt_params
{'verbose': 10,
'n_estimators': 30,
'max_features': 'sqrt',
'max_depth': 10,
'criterion': 'entropy'}
[148] bt_score
0.9916513275081691
[149] model=RandomForestClassifier(verbose=10,n_estimators=120,max_features='log2',max_depth=10,criterion='entropy')
RCV.fit(x_train,y_train)

```
Help  Save failed

+ Code  + Text

[150] [Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
      [Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.0s finished

[151] RFC=accuracy_score(y_test,y_predict_rfc)
      RFC

      0.9666221628838452

[152] import pickle
      pickle.dump(RCV,open('flight.pkl','wb'))

[153] from flask import Flask,request,render_template
      import numpy as np
      import pandas as pd
      import pickle
      import os

[154] model=pickle.load(open('flight.pkl','rb'))
```

```
fflightdata.ipynb
File  Edit  View  Insert  Runtime  Tools  Help  Last saved at 11:22 PM

+ Code  + Text

[221] app.route('/')
      def home():
          return render_template("index.html")
      app.route('/prediction',methods=['POST'])

      <function flask.scaffold.Scaffold.route.<locals>.decorator(f: ~T_route) -> ~T_route>

[223] 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
```

```
194] 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=="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,dayofmonth,dayofweek,origin1,origin2,origin3,origin4,origin5,destination1,destination2,destination3,destination4,destination5],
        [name2,month2,dayofmonth2,dayofweek2,origin1_2,origin2_2,origin3_2,origin4_2,origin5_2,destination1_2,destination2_2,destination3_2,destination4_2,destination5_2],
    ]
    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)
```


+ Code + Text
Connecting
[ ]
app = Flask(__name__)
if __name__ == '__main__':
app.run(debug=True)
* Serving Flask app '__main__'
* Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat


flight images hd - Google Search
fflightdata.ipynb - Colaboratory
colab.research.google.com/drive/1R3HhVTuFfs6HItjRR34U8-869SpXJUZd#scrollTo=Lc14YfHhyqoE
fflightdata.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
54
+ Code + Text
Connecting
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 == "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,dayofmonth,dayofweek,origin1,origin2,origin3,origin4,origin5,destination1,destination2,destination3,destination4,destination5]]
y_pred= model.predict(total)
print(y_pred)s
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)
images (8).jpg
images (7).jpg
download.jpg
images (6).jpg
images (5).jpg
Show all
12:39 PM
4/10/2023