Smart Internz

4/23/2023

Flight Delay Prediction For Aviation

Using Machine Learning

Flight Delay Prediction for Aviation Industry Using Machine Learning

GOVERNMENT ARTS AND SCIENCE

COLLEGE FOR WOMENS

SATHANKULAM

DEPARTMENT OF COMPUTER SCIENCE

NAME OF MENTER: Mrs. Saraswathy

Team Members :

Sowmiya.P	***-20202131506147***
Indhumathi.M	***-20202131506110***
Elizabeth Rani.M	***-20202131506106***
Thanalakshmi.K	***-20202131506154***

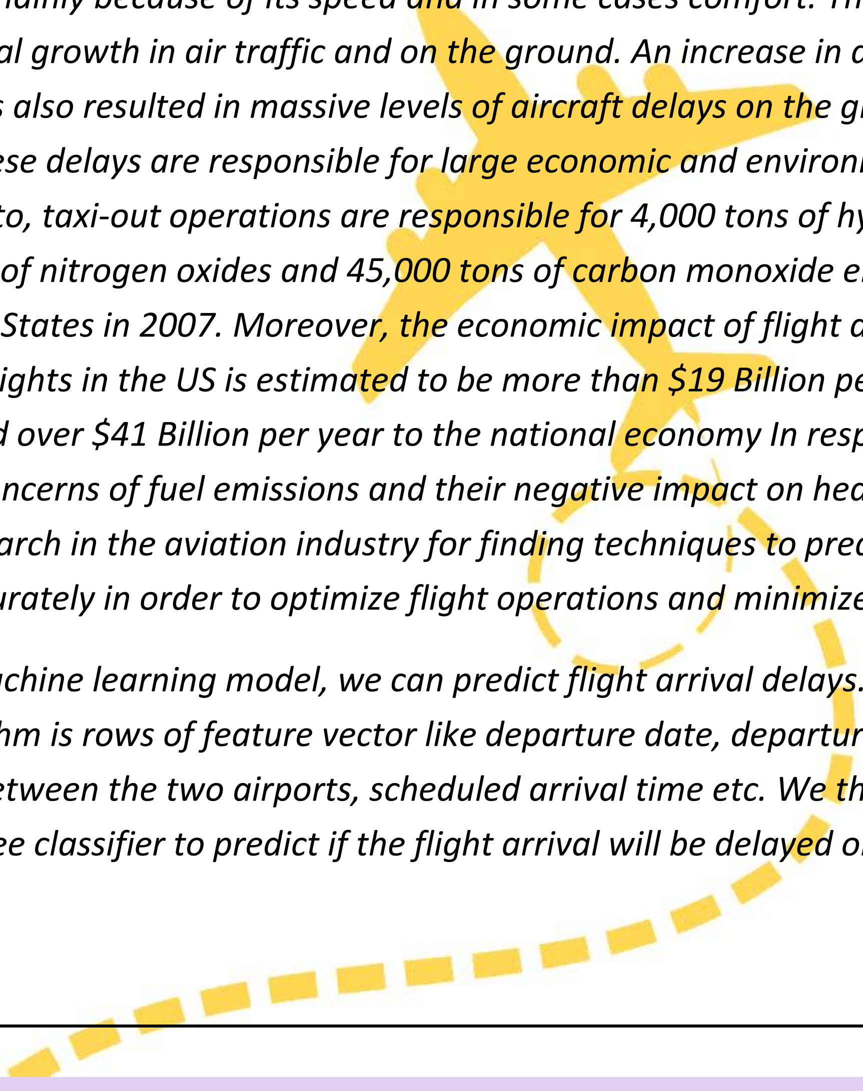
Project Report

INTRODUCTION

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.

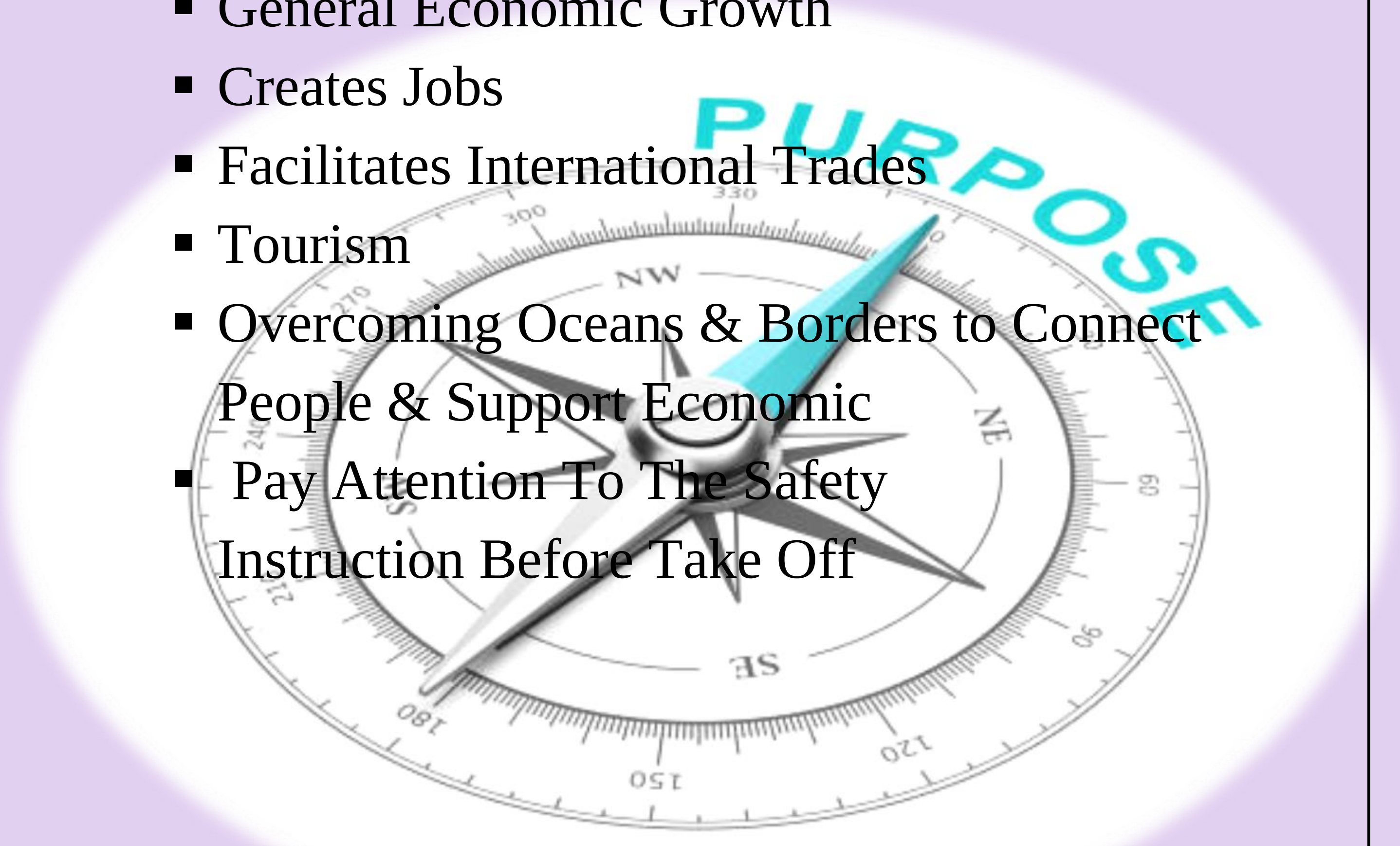
Using a machine learning model, we can predict flight arrival delays. The input to our algorithm is rows of feature vector like departure date, departure delay, distance between the two airports, scheduled arrival time etc. We then use decision tree classifier to predict if the flight arrival will be delayed or not. A flight

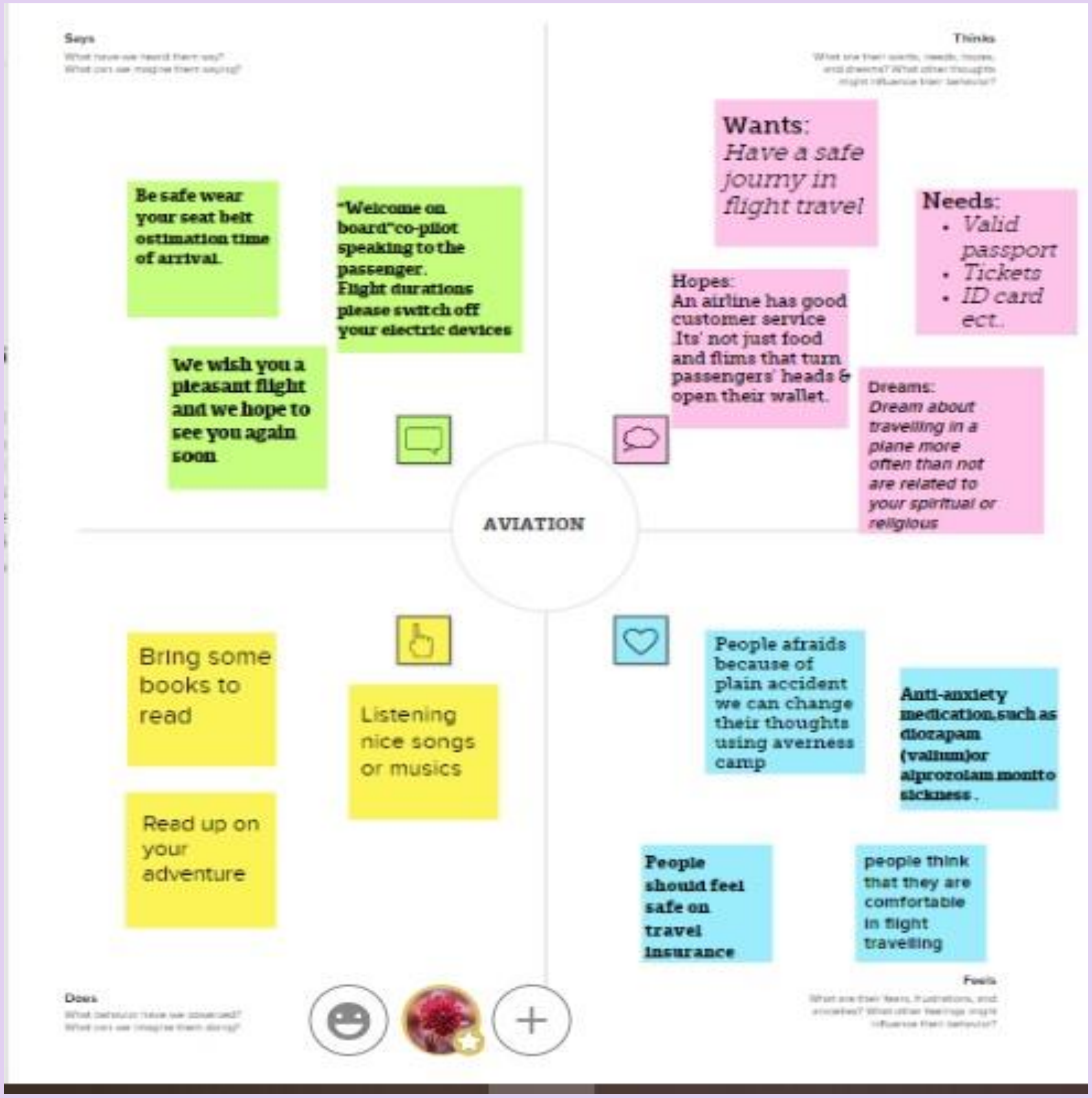


is delayed when difference between scheduled and actual arrival times is greater than 15 minutes. Furthermore, we compare decision tree classifier with logistic regression and a simple neural network for various figures of merit. Finally, it will be integrated to web based application

Purpose :

- Time for Fueling
- Boarding passenger
- Aircraft Cleaning
- Air Ticket Cleaning
- General Economic Growth
- Creates Jobs
- Facilitates International Trades
- Tourism
- Overcoming Oceans & Borders to Connect People & Support Economic
- Pay Attention To The Safety Instruction Before Take Off





Says

What have we heard them say?
What can we imagine them saying?

Be safe wear your seat belt estimation time of arrival.

"Welcome on board"co-pilot speaking to the passenger. Flight durations please switch off your electric devices

We wish you a pleasant flight and we hope to see you again soon

Thinks

What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?

Wants:
Have a safe journy in flight travel

Needs:
- *Valid passport*
- *Tickets*
- *ID card ect..*

Hopes:
An airline has good customer service .Its' not just food and flims that turn passengers' heads & open their wallet.

Dreams:
Dream about travelling in a plane more often than not are related to your spiritual or religious

AVIATION

Does

What behavior have we observed?
What can we imagine them doing?

Bring some books to read

Read up on your adventure

Listening nice songs or musics

Feels

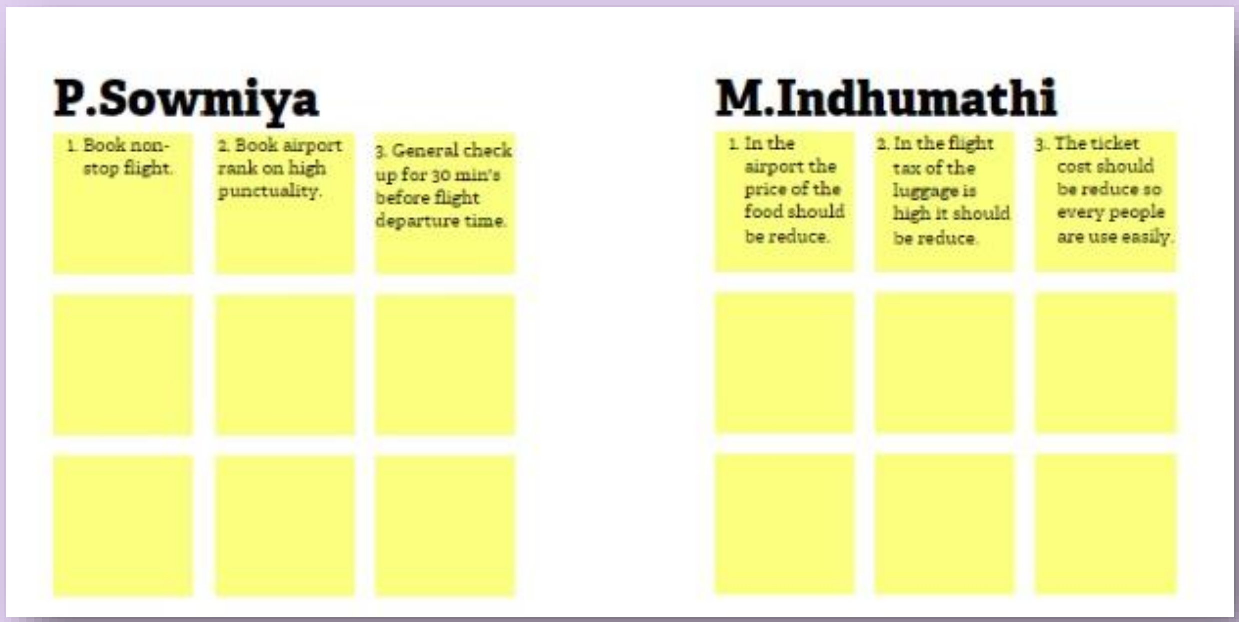
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

People afraids because of plain accident we can change their thoughts using averness camp

Anti-anxiety medication.such as diozapam (valium)or alprozolam monit to sickness .

People should feel safe on travel insurance

people think that they are comfortable in flight travelling

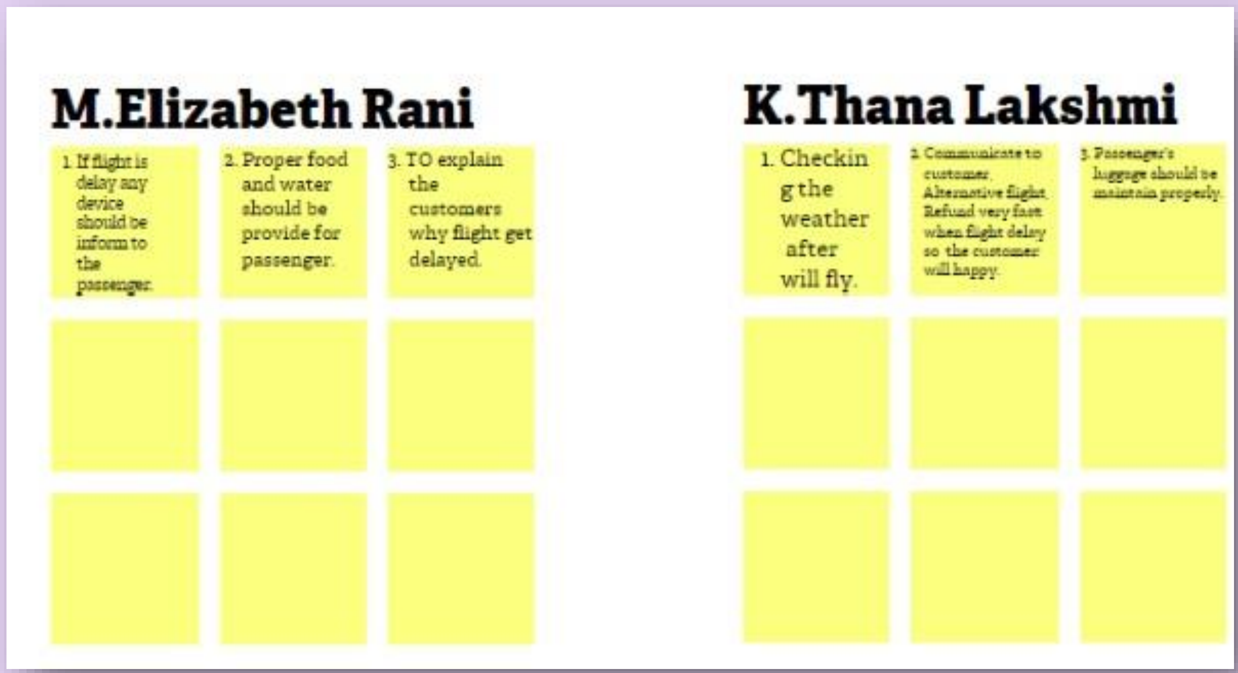


P.Sowmiya

1. Book non-stop flight.
2. Book airport rank on high punctuality.
3. General check up for 30 min's before flight departure time.

M.Indhumathi

1. In the airport the price of the food should be reduce.
2. In the flight tax of the luggage is high it should be reduce.
3. The ticket cost should be reduce so every people are use easily.



M.Elizabeth Rani

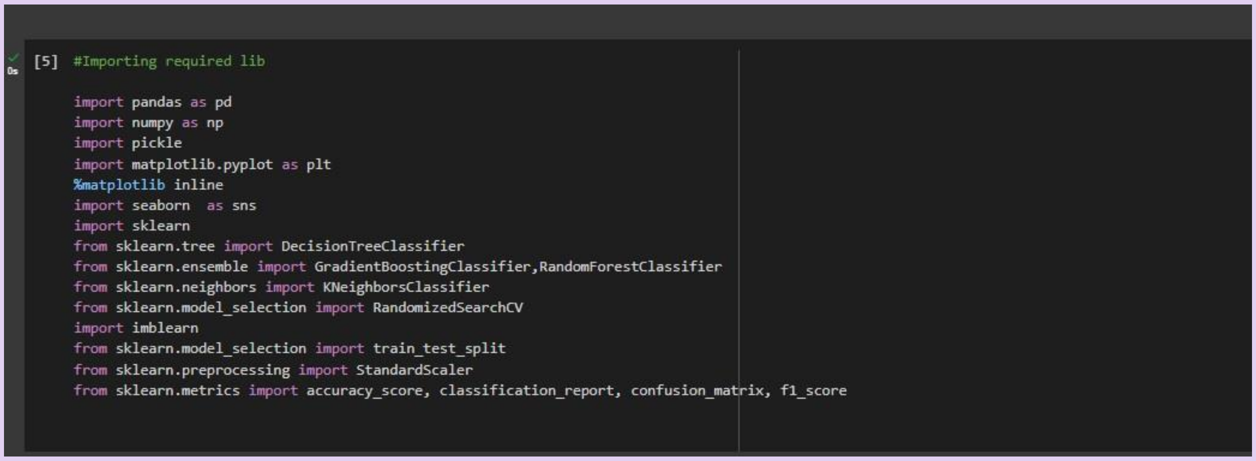
1. If flight is delay any device should be inform to the passenger.
2. Proper food and water should be provide for passenger.
3. TO explain the customers why flight get delayed.

K.Thana Lakshmi

1. Checking the weather after will fly.
2. Communicate to customer, Alternative flight, Refund very fast when flight delay so the customer will happy.
3. Passenger's luggage should be maintain properly.

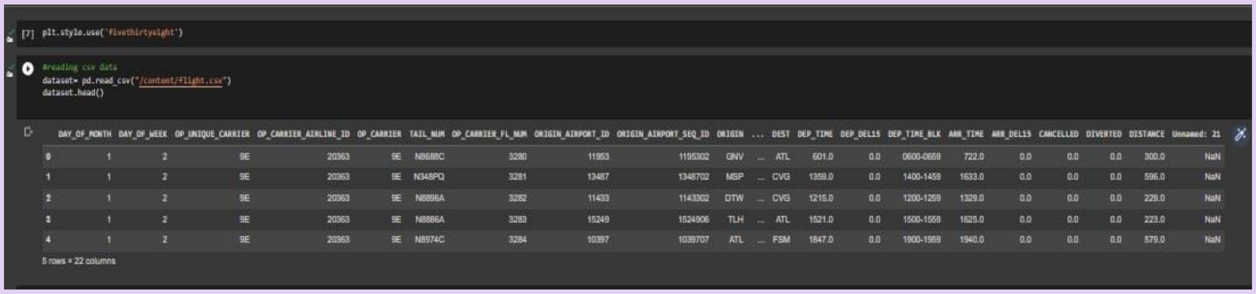
RESULTS:

Importing the libraries:



```
[5]  #Importing required lib

     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
```

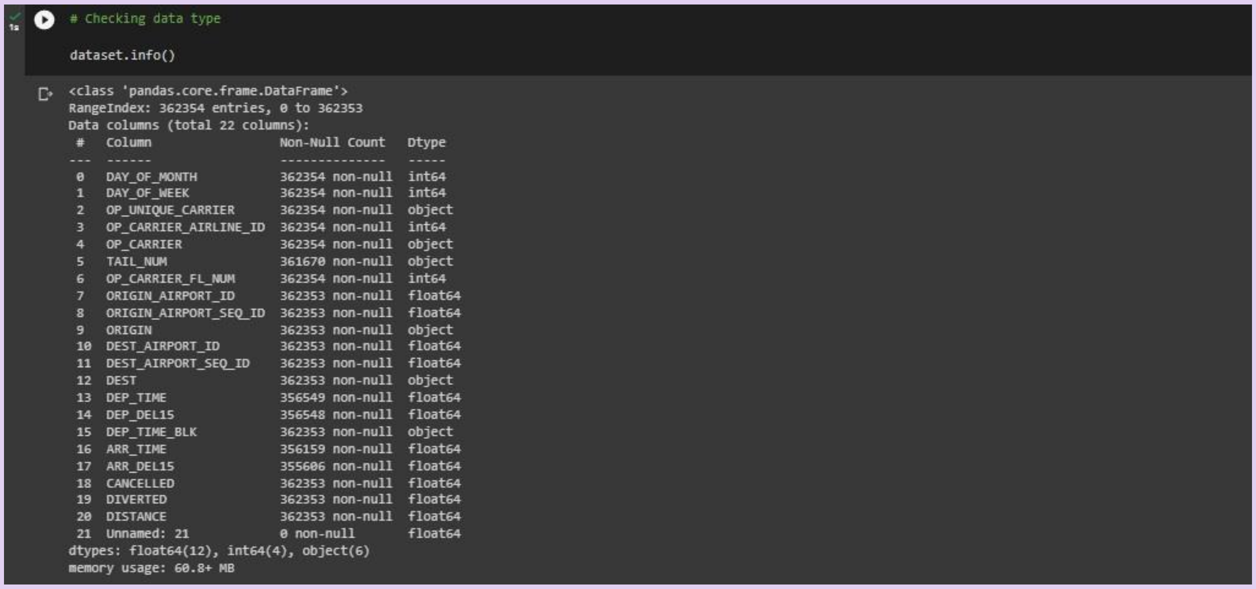
Read the Dataset:



```
[7]  plt.style.use('fivethirtyeight')

     #reading csv data
     dataset= pd.read_csv("/content/flight.csv")
     dataset.head()
```

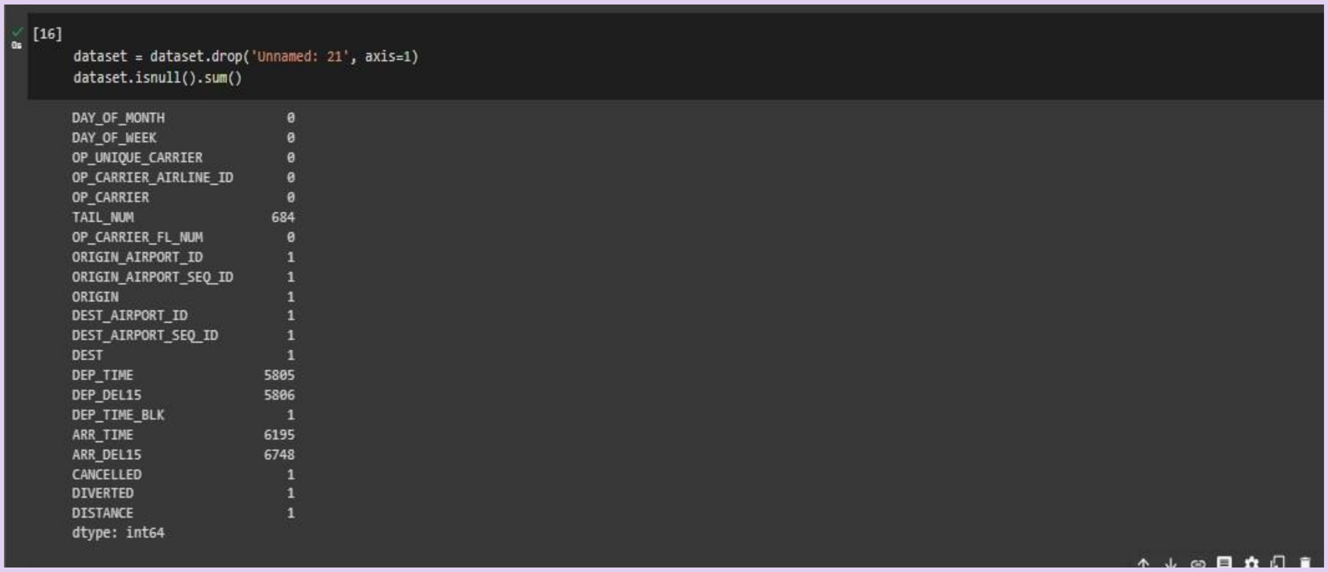
```
   DAY_OF_MONTH  DAY_OF_WEEK  OP_UNIQUE_CARRIER  OP_CARRIER_AIRLINE_ID  OP_CARRIER  TAIL_NUM  OP_CARRIER_FL_NUM  ORIGIN_AIRPORT_ID  ORIGIN_AIRPORT_SEQ_ID  ORIGIN  ...  DEST  DEP_TIME  DEP_DEL15  DEP_TIME_BLK  ARR_TIME  ARR_DEL15  CANCELLED  DIVERTED  DISTANCE  Unnamed: 21
0             1            2                 9E                  20363          9E    N8688C               3280              11953                1195302     GNV  ...   ATL     601.0        0.0     0600-0659     722.0        0.0        0.0       0.0     300.0          NaN
1             1            2                 9E                  20363          9E    N348PQ               3281              13487                1348702     MSP  ...   CVG    1359.0        0.0     1400-1459    1633.0        0.0        0.0       0.0     596.0          NaN
2             1            2                 9E                  20363          9E    N8896A               3282              11433                1143302     DTW  ...   CVG    1215.0        0.0     1200-1259    1329.0        0.0        0.0       0.0     229.0          NaN
3             1            2                 9E                  20363          9E    N8886A               3283              15249                1524906     TLH  ...   ATL    1521.0        0.0     1500-1559    1625.0        0.0        0.0       0.0     223.0          NaN
4             1            2                 9E                  20363          9E    N8974C               3284              10397                1039707     ATL  ...   FSM    1847.0        0.0     1900-1959    1940.0        0.0        0.0       0.0     579.0          NaN

5 rows × 22 columns
```

Handling missing values:



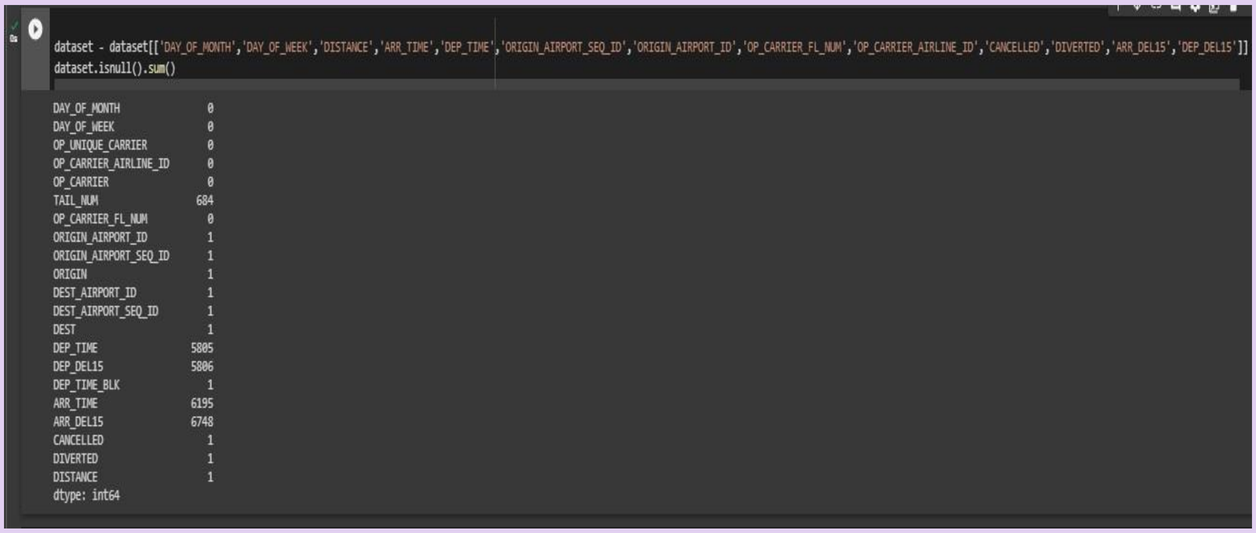
```
# Checking data type

dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 362354 entries, 0 to 362353
Data columns (total 22 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   DAY_OF_MONTH           362354 non-null  int64  
 1   DAY_OF_WEEK            362354 non-null  int64  
 2   OP_UNIQUE_CARRIER      362354 non-null  object 
 3   OP_CARRIER_AIRLINE_ID  362354 non-null  int64  
 4   OP_CARRIER             362354 non-null  object 
 5   TAIL_NUM               361670 non-null  object 
 6   OP_CARRIER_FL_NUM      362354 non-null  int64  
 7   ORIGIN_AIRPORT_ID      362353 non-null  float64
 8   ORIGIN_AIRPORT_SEQ_ID  362353 non-null  float64
 9   ORIGIN                 362353 non-null  object 
 10  DEST_AIRPORT_ID        362353 non-null  float64
 11  DEST_AIRPORT_SEQ_ID    362353 non-null  float64
 12  DEST                   362353 non-null  object 
 13  DEP_TIME               356549 non-null  float64
 14  DEP_DEL15              356548 non-null  float64
 15  DEP_TIME_BLK           362353 non-null  object 
 16  ARR_TIME               356159 non-null  float64
 17  ARR_DEL15              355606 non-null  float64
 18  CANCELLED              362353 non-null  float64
 19  DIVERTED               362353 non-null  float64
 20  DISTANCE               362353 non-null  float64
 21  Unnamed: 21            0 non-null       float64
dtypes: float64(12), int64(4), object(6)
memory usage: 60.8+ MB
```



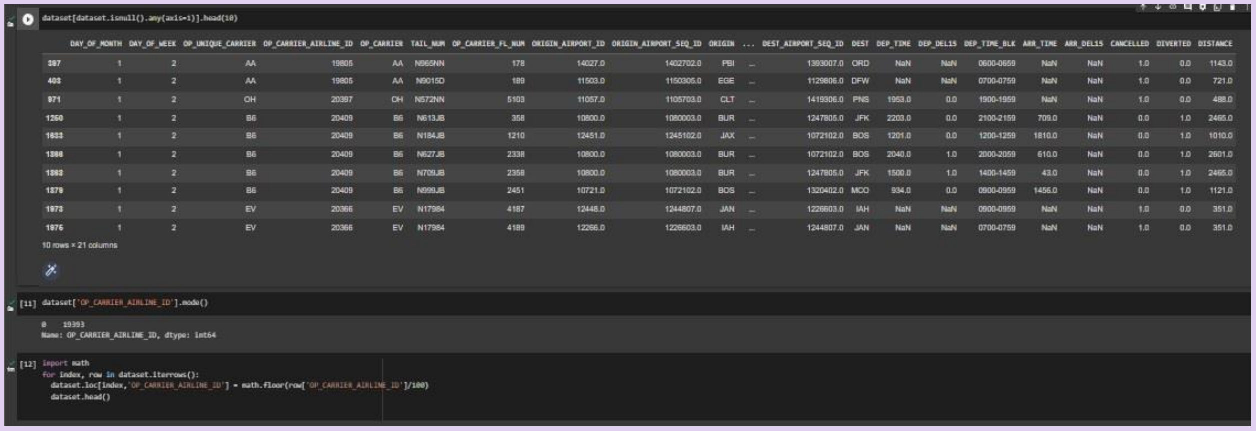
```
[16]
dataset = dataset.drop('Unnamed: 21', axis=1)
dataset.isnull().sum()
```

```
DAY_OF_MONTH               0
DAY_OF_WEEK                0
OP_UNIQUE_CARRIER          0
OP_CARRIER_AIRLINE_ID      0
OP_CARRIER                 0
TAIL_NUM                 684
OP_CARRIER_FL_NUM          0
ORIGIN_AIRPORT_ID          1
ORIGIN_AIRPORT_SEQ_ID      1
ORIGIN                     1
DEST_AIRPORT_ID            1
DEST_AIRPORT_SEQ_ID        1
DEST                       1
DEP_TIME                5805
DEP_DEL15               5806
DEP_TIME_BLK               1
ARR_TIME                6195
ARR_DEL15               6748
CANCELLED                  1
DIVERTED                   1
DISTANCE                   1
dtype: int64
```



```
dataset - dataset[['DAY_OF_MONTH','DAY_OF_WEEK','DISTANCE','ARR_TIME','DEP_TIME','ORIGIN_AIRPORT_SEQ_ID','ORIGIN_AIRPORT_ID','OP_CARRIER_FL_NUM','OP_CARRIER_AIRLINE_ID','CANCELLED','DIVERTED','ARR_DEL15','DEP_DEL15']]
dataset.isnull().sum()
```

```
DAY_OF_MONTH               0
DAY_OF_WEEK                0
OP_UNIQUE_CARRIER          0
OP_CARRIER_AIRLINE_ID      0
OP_CARRIER                 0
TAIL_NUM                 684
OP_CARRIER_FL_NUM          0
ORIGIN_AIRPORT_ID          1
ORIGIN_AIRPORT_SEQ_ID      1
ORIGIN                     1
DEST_AIRPORT_ID            1
DEST_AIRPORT_SEQ_ID        1
DEST                       1
DEP_TIME                5805
DEP_DEL15               5806
DEP_TIME_BLK               1
ARR_TIME                6195
ARR_DEL15               6748
CANCELLED                  1
DIVERTED                   1
DISTANCE                   1
dtype: int64
```



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

	DAY_OF_MONTH	DAY_OF_WEEK	OP_UNIQUE_CARRIER	OP_CARRIER_AIRLINE_ID	OP_CARRIER	TAIL_NUM	OP_CARRIER_FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN_AIRPORT_SEQ_ID	ORIGIN	...	DEST_AIRPORT_SEQ_ID	DEST	DEP_TIME	DEP_DEL15	DEP_TIME_BLK	ARR_TIME	ARR_DEL15	CANCELLED	DIVERTED	DISTANCE
387	1	2	AA	19805	AA	N965NN	178	14027.0	1402702.0	PBI	...	1393007.0	ORD	NaN	NaN	0600-0659	NaN	NaN	1.0	0.0	1143.0
403	1	2	AA	19805	AA	N9015D	189	11503.0	1150305.0	EGE	...	1129806.0	DFW	NaN	NaN	0700-0759	NaN	NaN	1.0	0.0	721.0
871	1	2	OH	20397	OH	N572NN	5103	11057.0	1105703.0	CLT	...	1419306.0	PNS	1953.0	0.0	1900-1959	NaN	NaN	1.0	0.0	488.0
1260	1	2	B6	20409	B6	N613JB	358	10800.0	1080003.0	BUR	...	1247805.0	JFK	2203.0	0.0	2100-2159	709.0	NaN	0.0	1.0	2465.0
1333	1	2	B6	20409	B6	N184JB	1210	12451.0	1245102.0	JAX	...	1072102.0	BOS	1201.0	0.0	1200-1259	1810.0	NaN	0.0	1.0	1010.0
1388	1	2	B6	20409	B6	N627JB	2338	10800.0	1080003.0	BUR	...	1072102.0	BOS	2040.0	1.0	2000-2059	610.0	NaN	0.0	1.0	2601.0
1393	1	2	B6	20409	B6	N709JB	2358	10800.0	1080003.0	BUR	...	1247805.0	JFK	1500.0	1.0	1400-1459	43.0	NaN	0.0	1.0	2465.0
1879	1	2	B6	20409	B6	N999JB	2451	10721.0	1072102.0	BOS	...	1320402.0	MCO	934.0	0.0	0900-0959	1456.0	NaN	0.0	1.0	1121.0
1973	1	2	EV	20366	EV	N17984	4187	12448.0	1244807.0	JAN	...	1226603.0	IAH	NaN	NaN	0900-0959	NaN	NaN	1.0	0.0	351.0
1976	1	2	EV	20366	EV	N17984	4189	12266.0	1226603.0	IAH	...	1244807.0	JAN	NaN	NaN	0700-0759	NaN	NaN	1.0	0.0	351.0

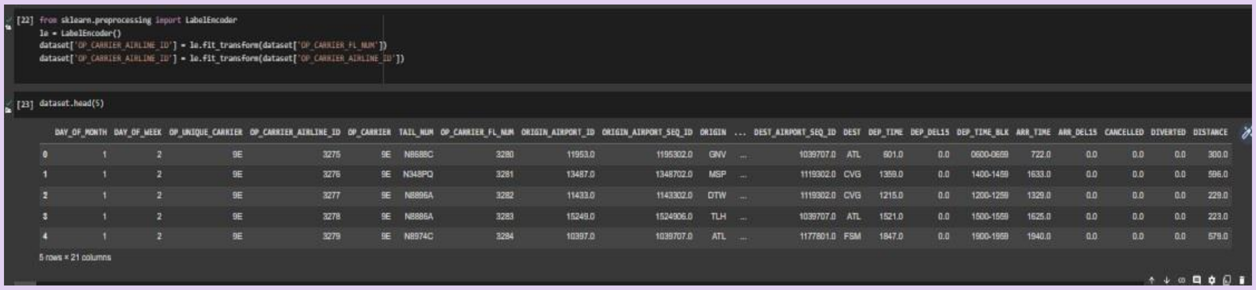
10 rows × 21 columns

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

```
0    19393
Name: OP_CARRIER_AIRLINE_ID, dtype: int64
```

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

Handling Categorical Values:

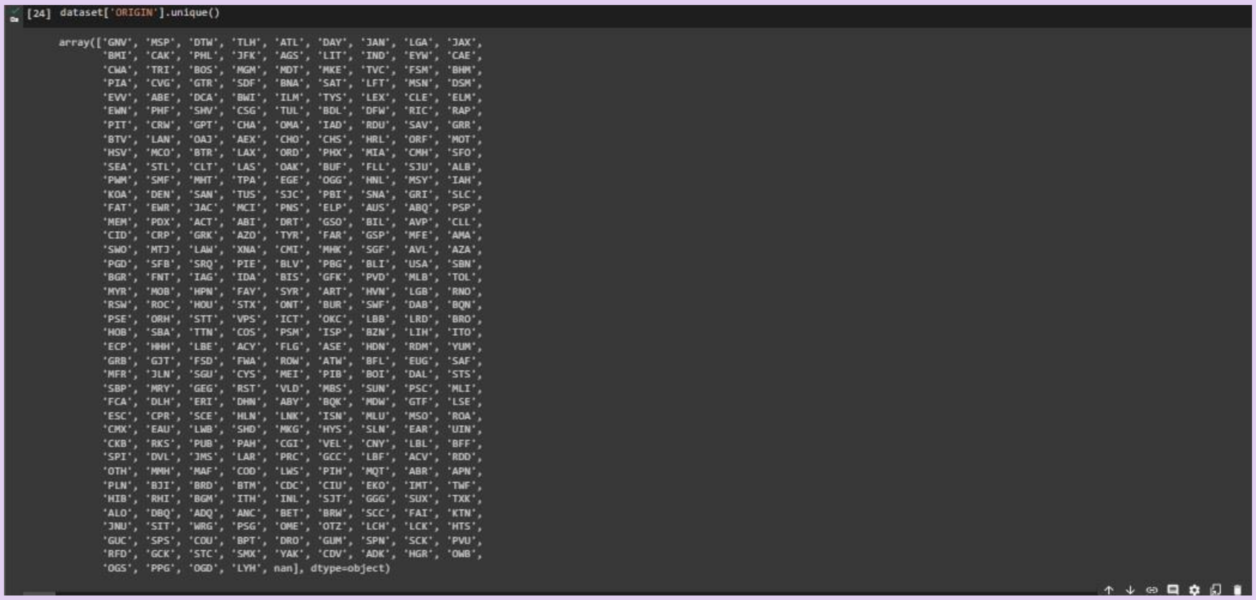


```
[22] from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     dataset['OP_CARRIER_AIRLINE_ID'] = le.fit_transform(dataset['OP_CARRIER_FL_NUM'])
     dataset['OP_CARRIER_AIRLINE_ID'] = le.fit_transform(dataset['OP_CARRIER_AIRLINE_ID'])
```

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

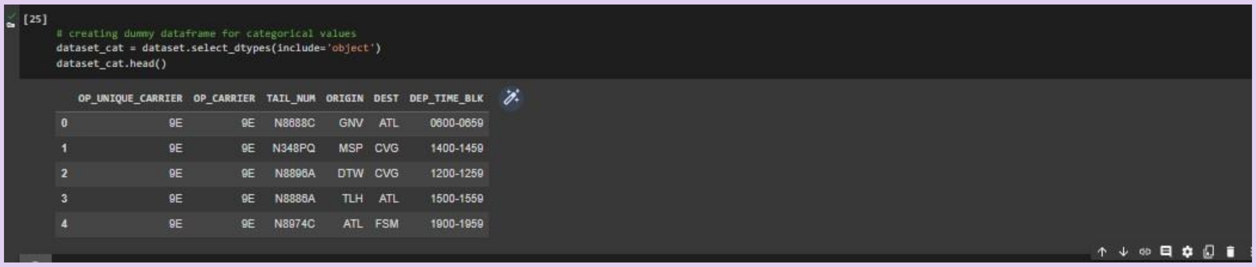
	DAY_OF_MONTH	DAY_OF_WEEK	OP_UNIQUE_CARRIER	OP_CARRIER_AIRLINE_ID	OP_CARRIER	TAIL_NUM	OP_CARRIER_FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN_AIRPORT_SEQ_ID	ORIGIN	...	DEST_AIRPORT_SEQ_ID	DEST	DEP_TIME	DEP_DEL15	DEP_TIME_BLK	ARR_TIME	ARR_DEL15	CANCELLED	DIVERTED	DISTANCE
0	1	2	9E	3275	9E	N8688C	3280	11953.0	1195302.0	GNV	...	1039707.0	ATL	601.0	0.0	0600-0659	722.0	0.0	0.0	0.0	300.0
1	1	2	9E	3276	9E	N348PQ	3281	13487.0	1348702.0	MSP	...	1119302.0	CVG	1359.0	0.0	1400-1459	1633.0	0.0	0.0	0.0	596.0
2	1	2	9E	3277	9E	N8896A	3282	11433.0	1143302.0	DTW	...	1119302.0	CVG	1215.0	0.0	1200-1259	1329.0	0.0	0.0	0.0	229.0
3	1	2	9E	3278	9E	N8886A	3283	15249.0	1524906.0	TLH	...	1039707.0	ATL	1521.0	0.0	1500-1559	1625.0	0.0	0.0	0.0	223.0
4	1	2	9E	3279	9E	N8974C	3284	10397.0	1039707.0	ATL	...	1177801.0	FSM	1847.0	0.0	1900-1959	1940.0	0.0	0.0	0.0	579.0

5 rows × 21 columns



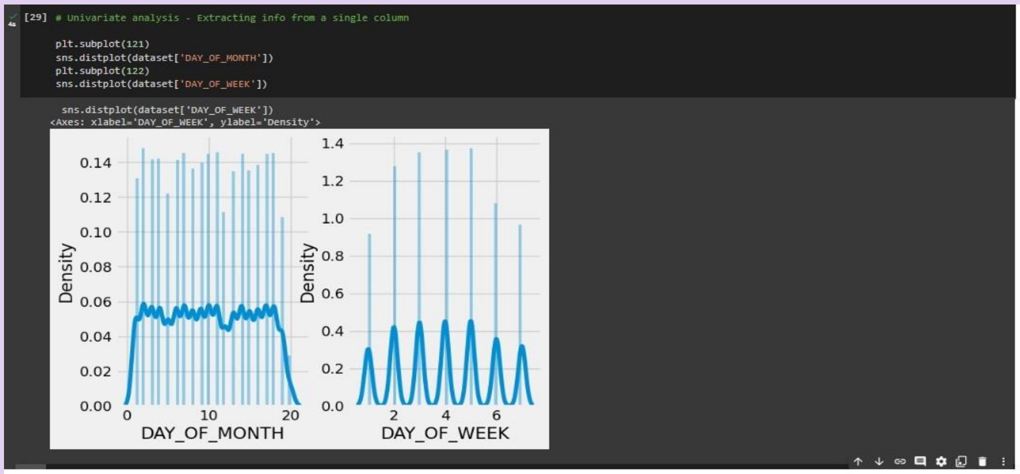
```
[24] dataset['ORIGIN'].unique()
```

```
array(['GNV', 'MSP', 'DTW', 'TLH', 'ATL', 'DAY', 'JAN', 'LGA', 'JAX',
       'BMI', 'CAK', 'PHL', 'JFK', 'AGS', 'LIT', 'IND', 'EYW', 'CAE',
       'CWA', 'TRI', 'BOS', 'MGM', 'MDT', 'MKE', 'TVC', 'FSM', 'BHM',
       'PIA', 'CVG', 'GTR', 'SDF', 'BNA', 'SAT', 'LFT', 'MSN', 'DSM',
       'EVV', 'ABE', 'DCA', 'BWI', 'ILM', 'TYS', 'LEX', 'CLE', 'ELM',
       'EWN', 'PHF', 'SHV', 'CSG', 'TUL', 'BDL', 'DFW', 'RIC', 'RAP',
       'PIT', 'CRW', 'GPT', 'CHA', 'OMA', 'IAD', 'RDU', 'SAV', 'GRR',
       'BTV', 'LAN', 'OAJ', 'AEX', 'CHO', 'CHS', 'HRL', 'ORF', 'MOT',
       'HSV', 'MCO', 'BTR', 'LAX', 'ORD', 'PHX', 'MIA', 'CMH', 'SFO',
       'SEA', 'STL', 'CLT', 'LAS', 'OAK', 'BUF', 'FLL', 'SJU', 'ALB',
       'PWM', 'SMF', 'MHT', 'TPA', 'EGE', 'OGG', 'HNL', 'MSY', 'IAH',
       'KOA', 'DEN', 'SAN', 'TUS', 'SJC', 'PBI', 'SNA', 'GRI', 'SLC',
       'FAT', 'EWR', 'JAC', 'MCI', 'PNS', 'ELP', 'AUS', 'ABQ', 'PSP',
       'MEM', 'PDX', 'ACT', 'ABI', 'DRT', 'GSO', 'BIL', 'AVP', 'CLL',
       'CID', 'CRP', 'GRK', 'AZO', 'TYR', 'FAR', 'GSP', 'MFE', 'AMA',
       'SWO', 'MTJ', 'LAW', 'XNA', 'CMI', 'MHK', 'SGF', 'AVL', 'AZA',
       'PGD', 'SFB', 'SRQ', 'PIE', 'BLV', 'PBG', 'BLI', 'USA', 'SBN',
       'BGR', 'FNT', 'IAG', 'IDA', 'BIS', 'GFK', 'PVD', 'MLB', 'TOL',
       'MYR', 'MOB', 'HPN', 'FAY', 'SYR', 'ART', 'HVN', 'LGB', 'RNO',
       'RSW', 'ROC', 'HOU', 'STX', 'ONT', 'BUR', 'SWF', 'DAB', 'BQN',
       'PSE', 'ORH', 'STT', 'VPS', 'ICT', 'OKC', 'LBB', 'LRD', 'BRO',
       'HOB', 'SBA', 'TTN', 'COS', 'PSM', 'ISP', 'BZN', 'LIH', 'ITO',
       'ECP', 'HHH', 'LBE', 'ACY', 'FLG', 'ASE', 'HDN', 'RDM', 'YUM',
       'GRB', 'GJT', 'FSD', 'FWA', 'ROW', 'ATW', 'BFL', 'EUG', 'SAF',
       'MFR', 'JLN', 'SGU', 'CYS', 'MEI', 'PIB', 'BOI', 'DAL', 'STS',
       'SBP', 'MRY', 'GEG', 'RST', 'VLD', 'MBS', 'SUN', 'PSC', 'MLI',
       'FCA', 'DLH', 'ERI', 'DHN', 'ABY', 'BQK', 'MDW', 'GTF', 'LSE',
       'ESC', 'CPR', 'SCE', 'HLN', 'LNK', 'ISN', 'MLU', 'MSO', 'ROA',
       'CMX', 'EAU', 'LWB', 'SHD', 'MKG', 'HYS', 'SLN', 'EAR', 'UIN',
       'CKB', 'RKS', 'PUB', 'PAH', 'CGI', 'VEL', 'CNY', 'LBL', 'BFF',
       'SPI', 'DVL', 'JMS', 'LAR', 'PRC', 'GCC', 'LBF', 'ACV', 'RDD',
       'OTH', 'MMH', 'MAF', 'COD', 'LWS', 'PIH', 'MQT', 'ABR', 'APN',
       'PLN', 'BJI', 'BRD', 'BTM', 'CDC', 'CIU', 'EKO', 'IMT', 'TWF',
       'HIB', 'RHI', 'BGM', 'ITH', 'INL', 'SJT', 'GGG', 'SUX', 'TXK',
       'ALO', 'DBQ', 'ADQ', 'ANC', 'BET', 'BRW', 'SCC', 'FAI', 'KTN',
       'JNU', 'SIT', 'WRG', 'PSG', 'OME', 'OTZ', 'LCH', 'LCK', 'HTS',
       'GUC', 'SPS', 'COU', 'BPT', 'DRO', 'GUM', 'SPN', 'SCK', 'PVU',
       'RFD', 'GCK', 'STC', 'SMX', 'YAK', 'CDV', 'ADK', 'HGR', 'OWB',
       'OGS', 'PPG', 'OGD', 'LYH', nan], dtype=object)
```



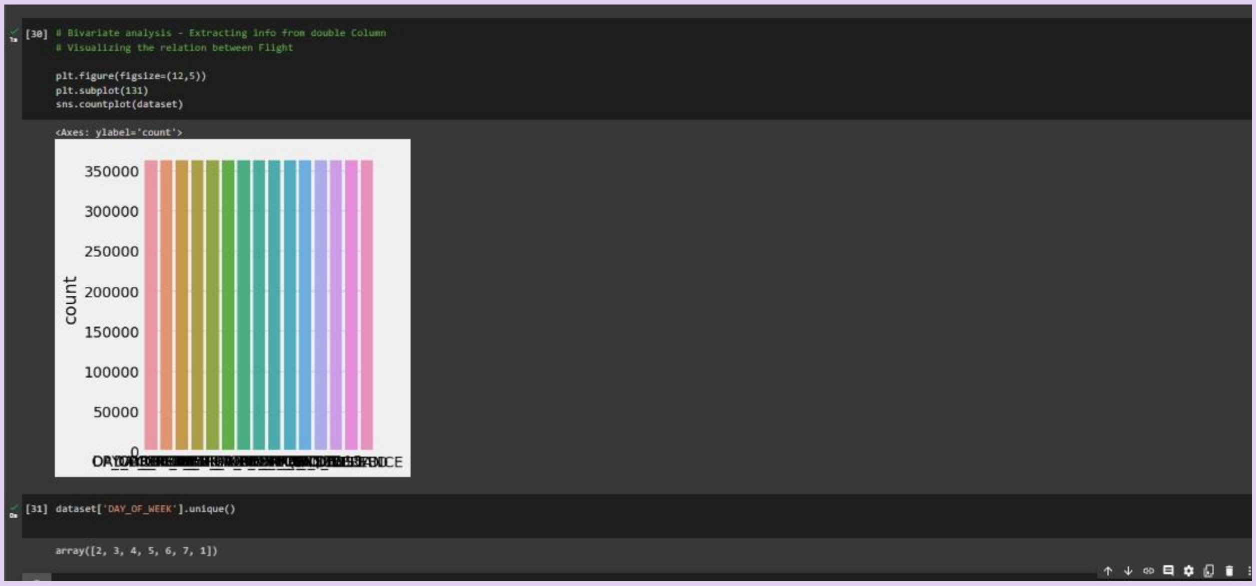
```
[25]
# creating dummy dataframe for categorical values
dataset_cat = dataset.select_dtypes(include='object')
dataset_cat.head()
```

	OP_UNIQUE_CARRIER	OP_CARRIER	TAIL_NUM	ORIGIN	DEST	DEP_TIME_BLK
0	9E	9E	N8688C	GNV	ATL	0600-0659
1	9E	9E	N348PQ	MSP	CVG	1400-1459
2	9E	9E	N8896A	DTW	CVG	1200-1259
3	9E	9E	N8886A	TLH	ATL	1500-1559
4	9E	9E	N8974C	ATL	FSM	1900-1959



```
[29] # Univariate analysis - Extracting info from a single column

plt.subplot(121)
sns.distplot(dataset['DAY_OF_MONTH'])
plt.subplot(122)
sns.distplot(dataset['DAY_OF_WEEK'])
```

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

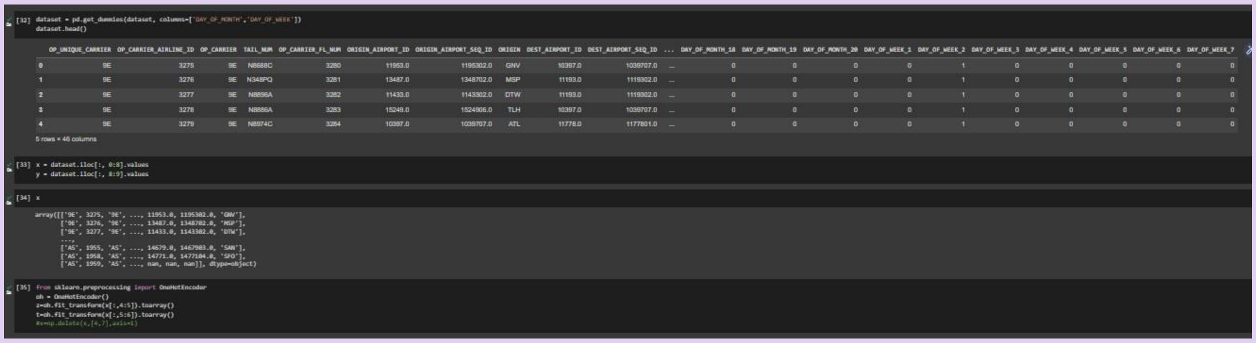


```
[30] # Bivariate analysis - Extracting info from double Column
     # Visualizing the relation between Flight

     plt.figure(figsize=(12,5))
     plt.subplot(131)
     sns.countplot(dataset)
```

```
<Axes: ylabel='count'>
```

```
[31] dataset['DAY_OF_WEEK'].unique()
```

```
array([2, 3, 4, 5, 6, 7, 1])
```



```
[32] dataset = pd.get_dummies(dataset, columns=['DAY_OF_MONTH','DAY_OF_WEEK'])
     dataset.head()
```

	OP_UNIQUE_CARRIER	OP_CARRIER_AIRLINE_ID	OP_CARRIER	TAIL_NUM	OP_CARRIER_FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN_AIRPORT_SEQ_ID	ORIGIN	DEST_AIRPORT_ID	DEST_AIRPORT_SEQ_ID	...	DAY_OF_MONTH_18	DAY_OF_MONTH_19	DAY_OF_MONTH_20	DAY_OF_WEEK_1	DAY_OF_WEEK_2	DAY_OF_WEEK_3	DAY_OF_WEEK_4	DAY_OF_WEEK_5	DAY_OF_WEEK_6	DAY_OF_WEEK_7
0	9E	3275	9E	N8688C	3280	11953.0	1195302.0	GNV	10397.0	1039707.0	...	0	0	0	0	1	0	0	0	0	0
1	9E	3276	9E	N348PQ	3281	13487.0	1348702.0	MSP	11193.0	1119302.0	...	0	0	0	0	1	0	0	0	0	0
2	9E	3277	9E	N8896A	3282	11433.0	1143302.0	DTW	11193.0	1119302.0	...	0	0	0	0	1	0	0	0	0	0
3	9E	3278	9E	N8886A	3283	15249.0	1524906.0	TLH	10397.0	1039707.0	...	0	0	0	0	1	0	0	0	0	0
4	9E	3279	9E	N8974C	3284	10397.0	1039707.0	ATL	11778.0	1177801.0	...	0	0	0	0	1	0	0	0	0	0

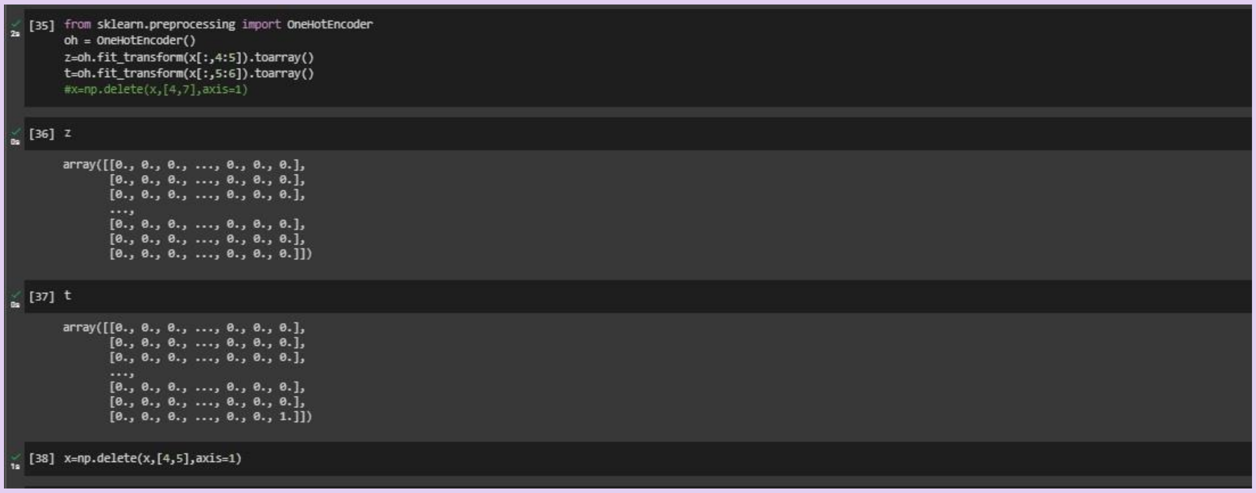
5 rows × 46 columns

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

```
[34] x
```

```
array([['9E', 3275, '9E', ..., 11953.0, 1195302.0, 'GNV'],
       ['9E', 3276, '9E', ..., 13487.0, 1348702.0, 'MSP'],
       ['9E', 3277, '9E', ..., 11433.0, 1143302.0, 'DTW'],
       ...,
       ['AS', 1955, 'AS', ..., 14679.0, 1467903.0, 'SAN'],
       ['AS', 1958, 'AS', ..., 14771.0, 1477104.0, 'SFO'],
       ['AS', 1959, 'AS', ..., nan, nan, nan]], dtype=object)
```

```
[35] 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,[4,7],axis=1)
```



```
[35] 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,[4,7],axis=1)
```

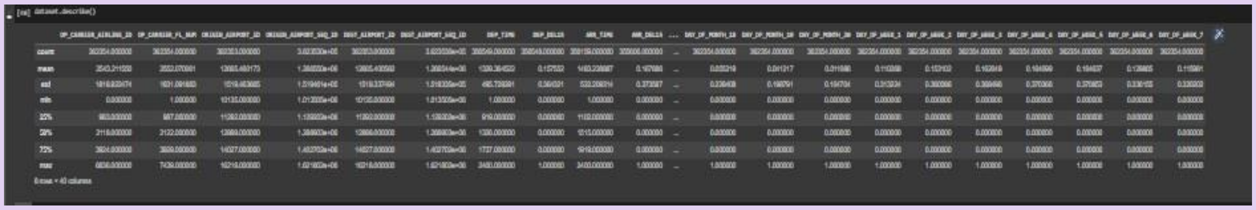
```
[36] z

     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.]])
```

```
[37] 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., 1.]])
```

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

Exploratory Data Analysis:

Descriptive statistical:



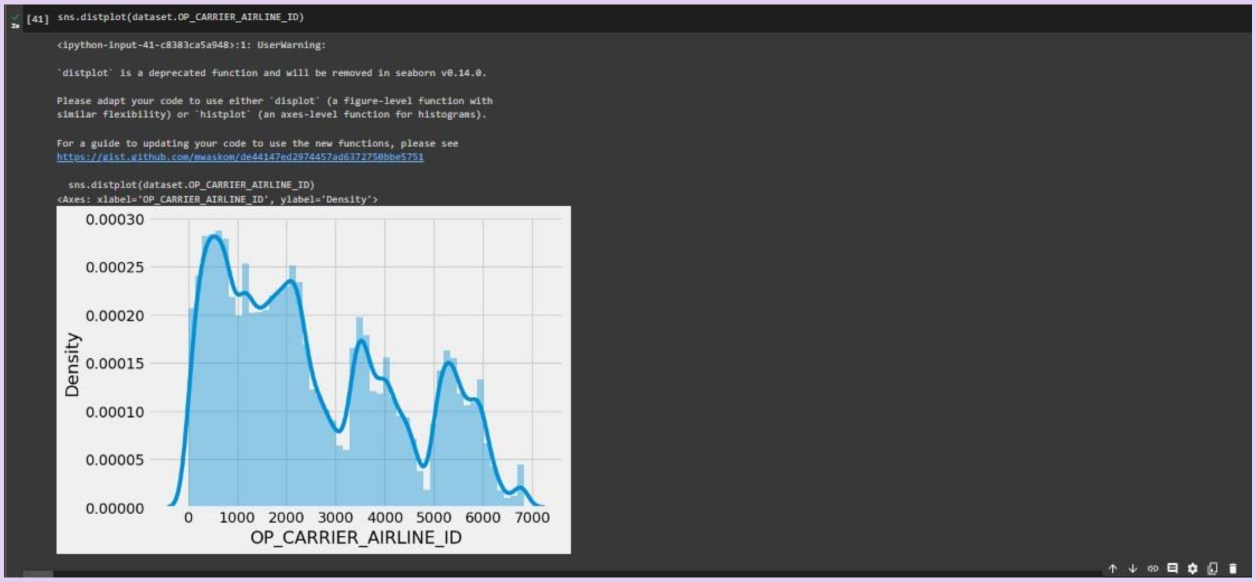
```
dataset.describe()
```

	OP_CARRIER_AIRLINE_ID	OP_CARRIER_FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN_AIRPORT_SEQ_ID	DEST_AIRPORT_ID	DEST_AIRPORT_SEQ_ID	DEP_TIME	DEP_DEL15	ARR_TIME	ARR_DEL15	...	DAY_OF_MONTH_29	DAY_OF_MONTH_30	DAY_OF_MONTH_31	DAY_OF_WEEK_1	DAY_OF_WEEK_2	DAY_OF_WEEK_3	DAY_OF_WEEK_4	DAY_OF_WEEK_5	DAY_OF_WEEK_6	DAY_OF_WEEK_7
count	362354.000000	362354.000000	362353.000000	3.623530e+05	362353.000000	3.623530e+05	358549.000000	358549.000000	358159.000000	358006.000000	...	362354.000000	362354.000000	362354.000000	362354.000000	362354.000000	362354.000000	362354.000000	362354.000000	362354.000000	362354.000000
mean	3543.211520	3553.070081	12665.483173	1.266550e+06	12665.430560	1.266544e+06	1330.364550	0.157532	1483.228887	0.167080	...	0.005219	0.011217	0.011986	0.110368	0.153102	0.162049	0.164699	0.164037	0.126605	0.115981
std	1816.822474	1831.091680	1519.463085	1.519461e+05	1519.337494	1.518330e+05	495.729391	0.364321	533.208314	0.373087	...	0.236408	0.100791	0.104704	0.313224	0.360066	0.368466	0.370366	0.370853	0.336155	0.320202
min	0.000000	1.000000	10135.000000	1.013505e+06	10135.000000	1.013505e+06	1.000000	0.000000	1.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	983.000000	987.000000	11292.000000	1.129202e+06	11292.000000	1.129202e+06	919.000000	0.000000	1102.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	2118.000000	2122.000000	12889.000000	1.288903e+06	12889.000000	1.288903e+06	1330.000000	0.000000	1515.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	3624.000000	3629.000000	14027.000000	1.402702e+06	14027.000000	1.402702e+06	1737.000000	0.000000	1919.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	6836.000000	7439.000000	16218.000000	1.621803e+06	16218.000000	1.621803e+06	2400.000000	1.000000	2400.000000	1.000000	...	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 40 columns

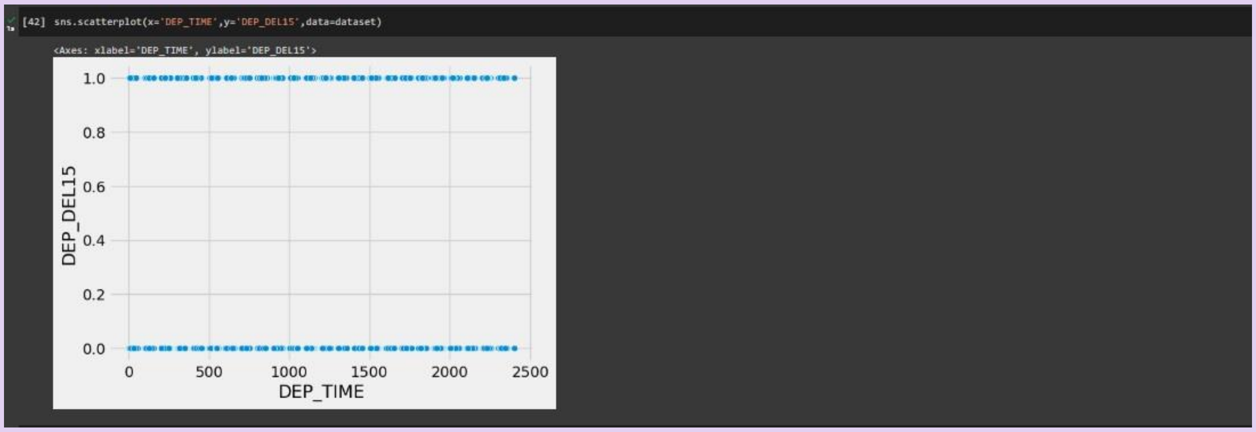
Visual analysis:

Univariate analysis:

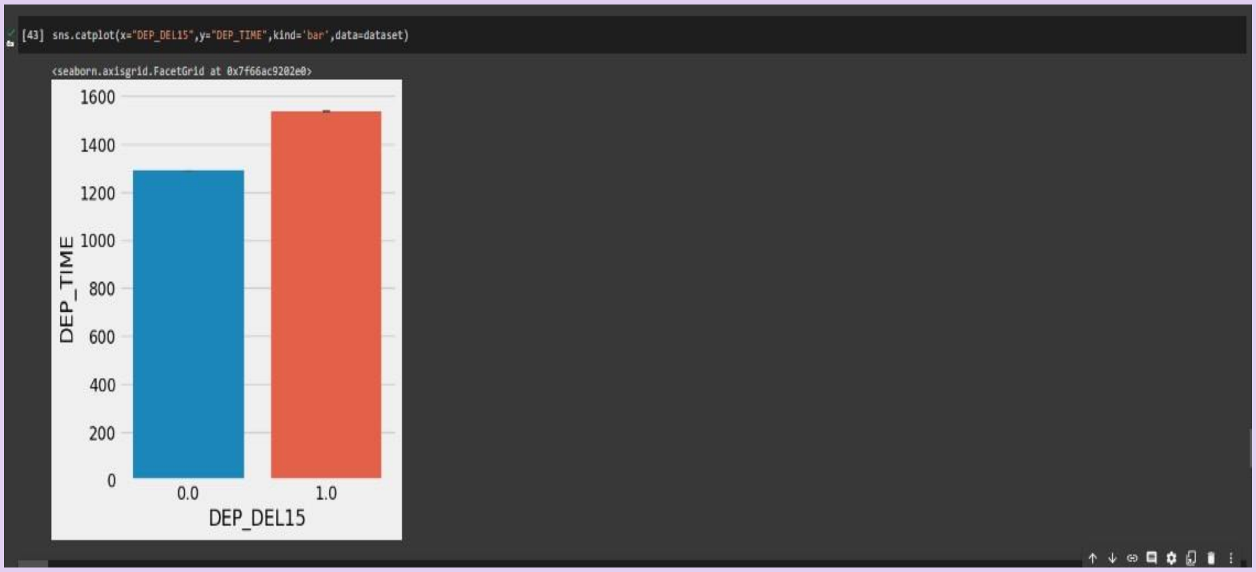


```
[41] sns.distplot(dataset.OP_CARRIER_AIRLINE_ID)

<ipython-input-41-c8383ca5a948>: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.OP_CARRIER_AIRLINE_ID)
<Axes: xlabel='OP_CARRIER_AIRLINE_ID', ylabel='Density'>
```

Bivariate analysis:

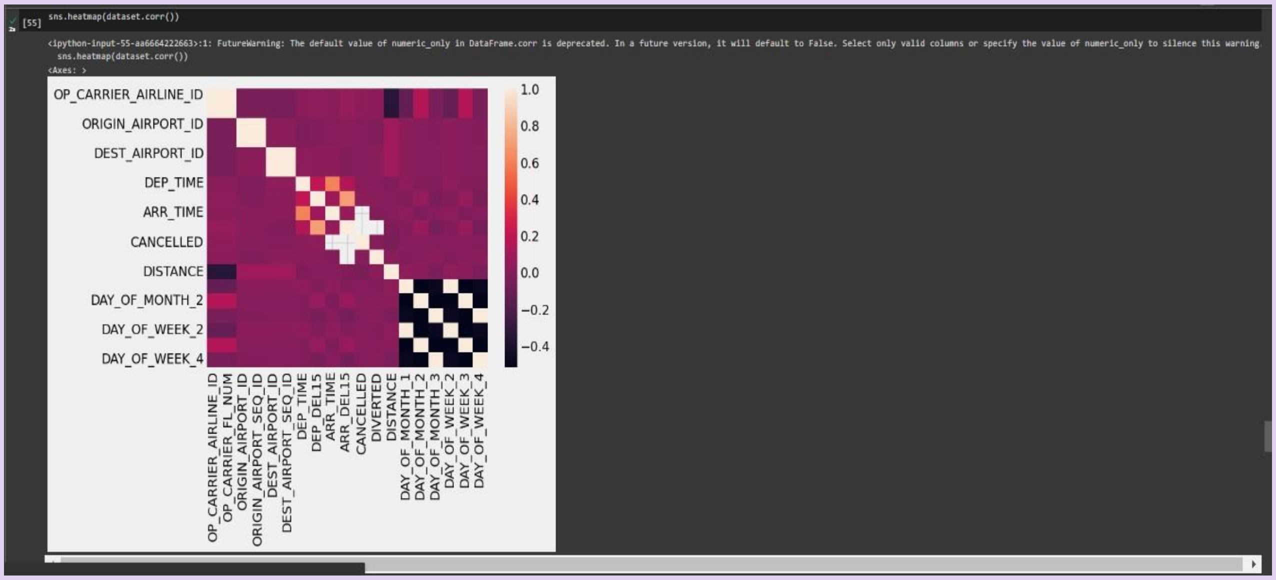


```
[42] sns.scatterplot(x='DEP_TIME',y='DEP_DEL15',data=dataset)

<Axes: xlabel='DEP_TIME', ylabel='DEP_DEL15'>
```



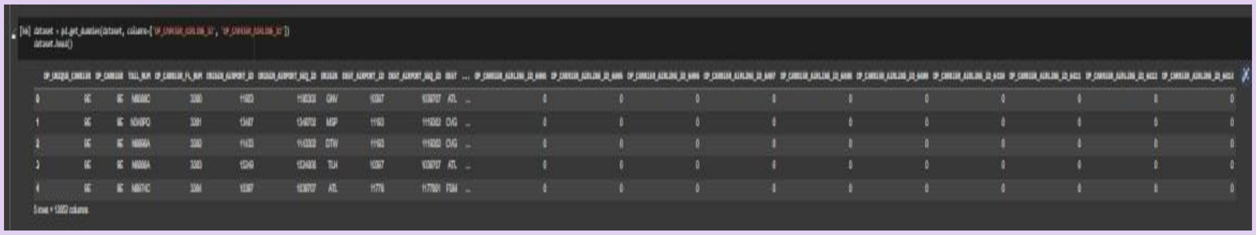
```
[43] sns.catplot(x="DEP_DEL15",y="DEP_TIME",kind='bar',data=dataset)

<seaborn.axisgrid.FacetGrid at 0x7f66ac9202e0>
```

Multivariate analysis:



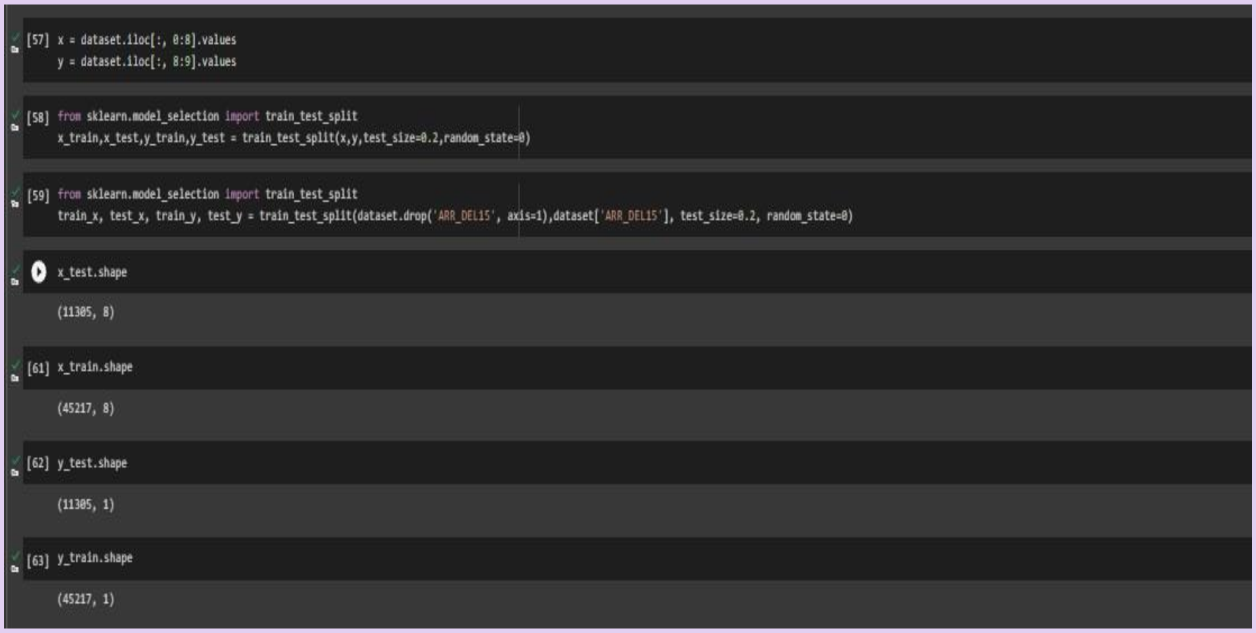
```
[55] sns.heatmap(dataset.corr())

<ipython-input-55-aa6664222663>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(dataset.corr())
<Axes: >
```

Splitting data into train and test:



```
[56] dataset = pd.get_dummies(dataset, columns=['OP_CARRIER_AIRLINE_ID', 'OP_CARRIER_AIRLINE_ID'])
     dataset.head()

   OP_UNIQUE_CARRIER OP_CARRIER TAIL_NUM OP_CARRIER_FL_NUM ORIGIN_AIRPORT_ID ORIGIN_AIRPORT_SEQ_ID ORIGIN DEST_AIRPORT_ID DEST_AIRPORT_SEQ_ID DEST ... OP_CARRIER_AIRLINE_ID_[illegible] OP_CARRIER_AIRLINE_ID_[illegible] OP_CARRIER_AIRLINE_ID_[illegible] OP_CARRIER_AIRLINE_ID_[illegible] OP_CARRIER_AIRLINE_ID_[illegible] OP_CARRIER_AIRLINE_ID_[illegible] OP_CARRIER_AIRLINE_ID_[illegible] OP_CARRIER_AIRLINE_ID_[illegible] OP_CARRIER_AIRLINE_ID_[illegible] OP_CARRIER_AIRLINE_ID_[illegible]
0                 9E         9E   N8688C              3280             11953               1195302    GNV           10397             1039707  ATL ...  0  0  0  0  0  0  0  0  0  0
1                 9E         9E   N348PQ              3281             13487               1348702    MSP           11193             1119302  CVG ...  0  0  0  0  0  0  0  0  0  0
2                 9E         9E   N8896A              3282             11433               1143302    DTW           11193             1119302  CVG ...  0  0  0  0  0  0  0  0  0  0
3                 9E         9E   N8886A              3283             15249               1524906    TLH           10397             1039707  ATL ...  0  0  0  0  0  0  0  0  0  0
4                 9E         9E   N8974C              3284             10397               1039707    ATL           11778             1177801  FSM ...  0  0  0  0  0  0  0  0  0  0

5 rows × 12052 columns
```

Model Building:



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

[58] 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)

[59] 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

     (11305, 8)

[61] x_train.shape

     (45217, 8)

[62] y_test.shape

     (11305, 1)

[63] y_train.shape

     (45217, 1)
```

Advantages :

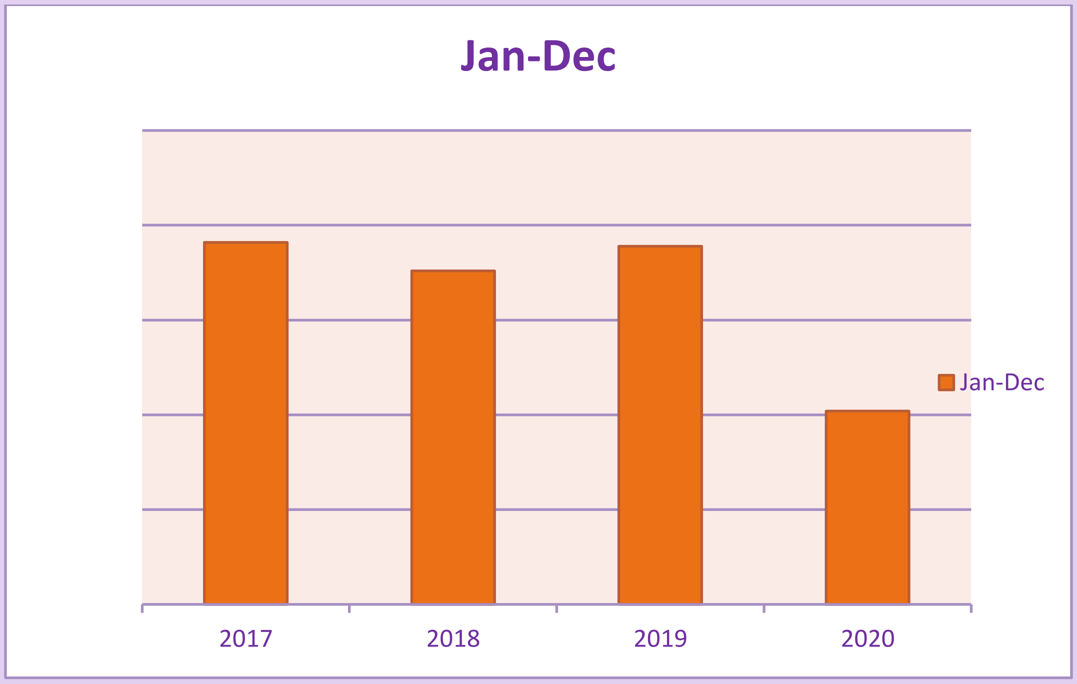
- Fastest type of travel.
- Good facilities at most airports refreshments/meals en route.
- Minimal check -in time for most domestic flights (within the UK).
- Most popular method of transport if going abroad .
- You are entitled to free-of-charge meals or refreshments.
- Reimbursement of your ticket and a return flight to your departure airport if you have a connecting flight.
- If your flight is delayed by more than 5 hours you decide not to travel and you are entitled to a full refund.
- Rerouting to your final destination.
- Rerouting at a later date under comparable transportation conditions.

- Customers can claim a refund of up to Rs 5,000.
- Customers are entitled for a full refund or re-booking onto an alternative Indigo flight at no additional cost subject to availability.

Disadvantages :

- **Financial losses.**
- **The dissatisfaction of passengers.**
- **Time losses.**
- **Loss of reputation .**
- **Bad business relations.**
- **Loss of demand by Passengers.**
- **Disagreement.**
- **Long check-in time required for some flight abroad.**
- **Decrease in efficiency, an increase in capital costs, reallocation of flight crews and aircraft, and additional crew expenses.**

Percentage of flight delay in year (2017-2020)

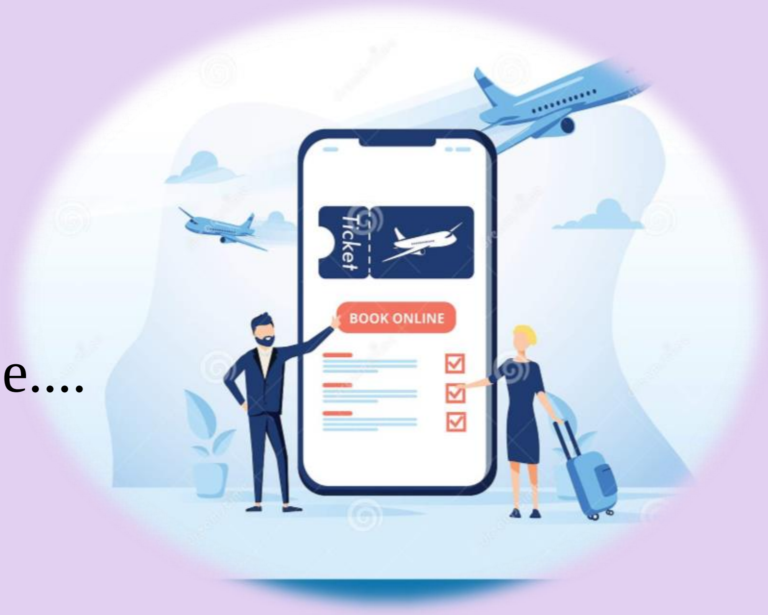


Application :

There are no federal laws requiring airlines to provide passengers with money or other compensation when their flights are delayed. Each airline has its own policies about what it will do for delayed passengers. If your flight is experiencing a long delay, ask airline staff if they will pay for meals or a hotel room.

To receive compensation for a flight delay or cancellation, you must make a claim with the airline in writing within 1 year of the incident date.

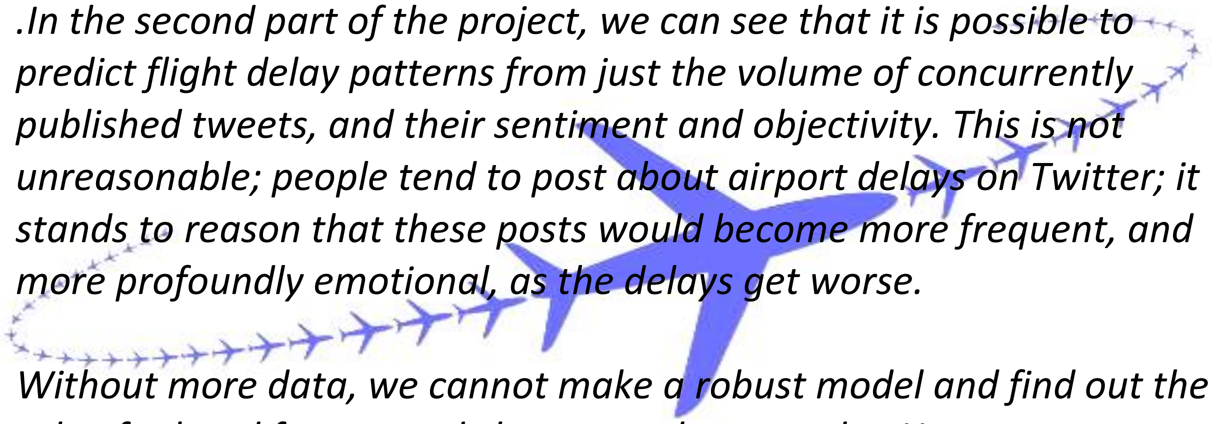
- Carrier Delay : Carrier delay is with in the control of the air carrier....
- Late Arrival Delay : Arrival delay at an airport due to the late arrival of the same aircraft at a previous airport....
- NAS Delay....
- Security Delay....
- Weather Delay....
- OPSNET Delay Cause....



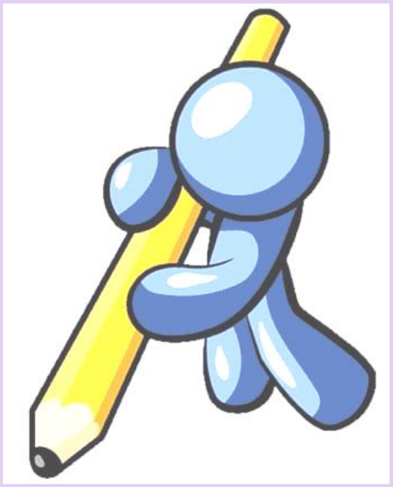
Conclusion :

In this project, we use flight data, weather, and demand data to predict flight departure delay. Our result shows that the Random Forest method yields the best performance compared to the SVM model. Somehow the SVM model is very time consuming and does not necessarily produce better results. In the end, our model correctly predicts 91% of the non-delayed flights. However, the delayed flights are only correctly predicted 41% of time.

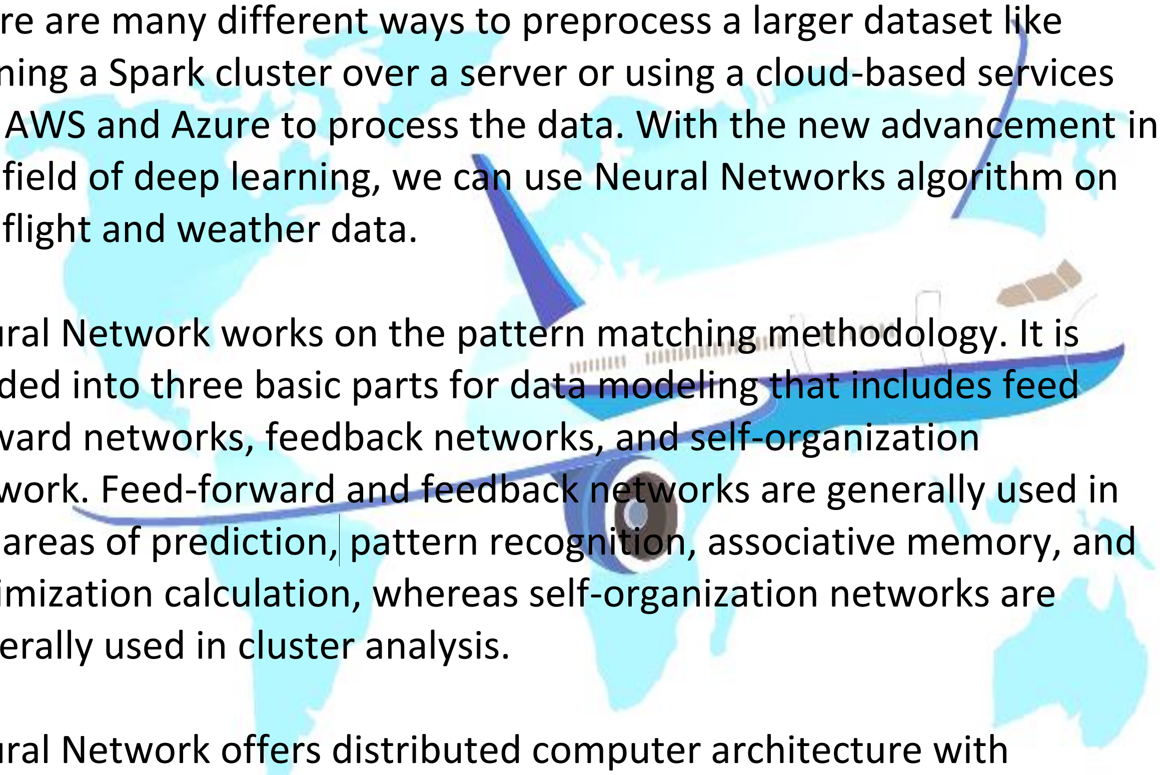
As a result, there can be additional features related to the causes of flight delay that are not yet discovered using our existing data sources .In the second part of the project, we can see that it is possible to predict flight delay patterns from just the volume of concurrently published tweets, and their sentiment and objectivity. This is not unreasonable; people tend to post about airport delays on Twitter; it stands to reason that these posts would become more frequent, and more profoundly emotional, as the delays get worse.



Without more data, we cannot make a robust model and find out the role of related factors and chance on these results. However, as a proof of concept, there is potential for these results. It may be possible to routinely use tweets to ascertain an understanding of concurrent airline delays and traffic patterns, which could be useful in a variety of circumstances.



Future scope :



This project is based on data analysis from year 2008. A large dataset is available from 1987-2008 but handling a bigger dataset requires a great amount of preprocessing and cleaning of the data. Therefore, the future work of this project includes incorporating a larger dataset. There are many different ways to preprocess a larger dataset like running a Spark cluster over a server or using a cloud-based services like AWS and Azure to process the data. With the new advancement in the field of deep learning, we can use Neural Networks algorithm on the flight and weather data.

Neural Network works on the pattern matching methodology. It is divided into three basic parts for data modeling that includes feed forward networks, feedback networks, and self-organization network. Feed-forward and feedback networks are generally used in the areas of prediction, pattern recognition, associative memory, and optimization calculation, whereas self-organization networks are generally used in cluster analysis.

Neural Network offers distributed computer architecture with important learning abilities to represent nonlinear relationships. Also, the scope of this project is very much confined to flight and weather data of United States, but we can include more countries like China, India, and Russia. Expanding the scope of this project, we can also add the flight data from international flights and not just restrict our self to the domestic flights.

SOURCES CODE:

Importing the libraries:

```
#Importing required lib

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 DecisionTreeClassifie
r
from sklearn.ensemble import GradientBoostin
gClassifier,RandomForestClassifier
from sklearn.neighbors import KNeighborsCla
ssifier
from sklearn.model_selection import Randomiz
edSearchCV
```

```
import imblearn
from sklearn.model_selection import train_test
_split
from sklearn.preprocessing import StandardSc
aler
from sklearn.metrics import accuracy_score, cl
assification_report, confusion_matrix, f1_score
```

Read the Dataset:

```
#reading csv data
dataset= pd.read_csv("/content/flight.csv")
dataset.head()
```

Handling missing values:

```
# Checking data type

dataset.info()

# Checking data type
```

```
dataset.info()

dataset -
 dataset[['DAY_OF_MONTH','DAY_OF_WEEK','DISTANCE','ARR_TIME','DEP_TIME','ORIGIN_AIRPORT_SEQ_ID','ORIGIN_AIRPORT_ID','OP_CARRIER_FL_NUM','OP_CARRIER_AIRLINE_ID','CANCELLED','DIVERTED','ARR_DEL15','DEP_DEL15']]
dataset.isnull().sum()


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

dataset['OP_CARRIER_AIRLINE_ID'].mode()
```

Handling Categorical Values:

```
import math
```

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

from sklearn.preprocessing import LabelEncod
er
le = LabelEncoder()
dataset['OP_CARRIER_AIRLINE_ID'] = le.fit
_transform(dataset['OP_CARRIER_FL_NUM'
])
dataset['OP_CARRIER_AIRLINE_ID'] = le.fit
_transform(dataset['OP_CARRIER_AIRLINE
_ID'])

dataset.head(5)

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

```
# creating dummy dataframe for categorical va
lues

dataset_cat = dataset.select_dtypes(include='ob
ject')
dataset_cat.head()

 # Univariate analysis -
 Extracting info from a single column

plt.subplot(121)
sns.distplot(dataset['DAY_OF_MONTH'])
plt.subplot(122)
sns.distplot(dataset['DAY_OF_WEEK'])


# Bivariate analysis -
 Extracting info from double Column
# Visualizing the relation between Flight

plt.figure(figsize=(12,5))
plt.subplot(131)
sns.countplot(dataset)

dataset['DAY_OF_WEEK'].unique()

dataset = pd.get_dummies(dataset, columns=['
DAY_OF_MONTH','DAY_OF_WEEK'])
dataset.head()
x = dataset.iloc[:, 0:8].values
y = dataset.iloc[:, 8:9].values


x

from sklearn.preprocessing import OneHotEnc
oder
oh = OneHotEncoder()
z=oh.fit_transform(x[:,4:5]).toarray()
t=oh.fit_transform(x[:,5:6]).toarray()
#x=np.delete(x,[4,7],axis=1)
```

z

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

Exploratory Data Analysis:

Descriptive statistical:

dataset.describe()

Visual analysis:

Univaiate analysis:

sns.distplot(dataset.OP_CARRIER_AIRLINE_ID)

Bivariate analysis:

sns.scatterplot(x='DEP_TIME',y='DEP_DEL15',data=dataset)

sns.catplot(x="DEP_DEL15",y="DEP_TIME",kind='bar',data=dataset)

Multivariate analysis:

```
sns.heatmap(dataset.corr())
```

Splitting data into train and test:

```
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),dataset['
ARR_DEL15'], test_size=0.2, random_state=0)

x_test.shape

x_train.shape

y_test.shape
```

```
y_train.shape
```

Scaling the data:

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

Model Building:

Decision Tree Classifier:

```
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)
```

Random forest model:

```
from sklearn.ensemble import RandomForestC
lassifier
rfc = RandomClassifier(n_estimators=10,criteri
on='entropy')


rfc.fit(x_train,y_train)


y_predict = rfc.predict(x_test)
```

ANN model:

```
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
’))

#Compiling the ANN model

classification.compile(optimizer=’adam,loss=’b
inarry_crossentropy’,metrics=[‘accuracy’])

#Training the model
```

classification.fit(x_train,y_train,bath_size=4,validation_split=0.2,epochs=100)

Test the model:

```
## Decision tree
y_pred =
classifie.predict([[129,99,1,0,0,1,0,1,1,1,0,1,1,1,
1,1]])
print(y_pred)
(y_pred)

## RandomForest

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')

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

y_pred


y_pred = (y_pred > 0.5)
y_pred


def predict_exit(sample_value):
#convert list to numpy array

sample_value = np.array(sample_value)

#Reshape because sample_value contrains only
1 record
sample_value = sample_value.reshape(1,-1)
#Feature scaling
sample_value = sc.transform(sample_value)
seturn classifier.predict(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’)
```

Creating Templates

1.Home.html

```
<html>

<head>
<title>Flight Delay Prediction</title>

<style type="text/csv">

@font-face {
font-family: myFirstFont;
src: url(font/Roboto-Regular.ttf);
/*font-weight: bold;*/
}
```

```
body
{
font-family: myFirstFont;
}
.sub_btn
{
background: green;
padding: 10px;
border-radius: 4px;
border: none;
margin-top: 30px;
width: 100%;
color: white;
font-size: 16px;
}
```

```
.main_section
{
width: 100%;
margin: auto;
text-align: center;
}

body
{
background-image: url("img.jpg");
/*height: 100%;*/

/*background: linear-gradient(rgb(193 196 225 /
80%), rgb(237 158 37 / 80%)), url(img.jpg);*/
background-position: center;
background-repeat: no-repeat;
background-size: cover;

}

#delay_result

{

width: 60%;

margin: auto;

letter-spacing: 0.8px;

line-height: 35px;

font-size: 17px;

}

.navbar

{

width: 100%;

height: 60px;

}

.navbar_ul
```

```
{
float: right;
}
.navbar_li
{
float: left;
background-color: royalblue;
padding: 10px;
margin-right: 10px;
list-style: none;
}
.nav-icon
{
color: white;
padding: 10px;
text-decoration: none;
}
</style>

</head>
<body>
<div class="main_section">
<div class="navbar">
<ul class="navbar_ul">
<li class="navbar_li"><a href="./home.html"
class="nav-icon">Home</a></li>
<li class="navbar_li"><a
href="./prediction.html" class="nav-
icon">Predict</a></li>
</ul>
</div>
<h1 >Flight Price Prediction</h1>
```

```
<div class="form_section">

<p id="delay_result">

The objective of this article is to predict flight
delay given the various parameters. Nowadays,
the number of people using flights has increased
significantly. That's why we will try to use
machine learning to solve this problem. This can
help airlines by predicting why flight delay  how
they can maintain. It can help customers to
predict future flight delay and plan their journey
accordingly.

</p>

</div>

</div>


</body>


</html>
```

2.Predict.html

```
<html>

<head>
<title>Flight Delay Prediction</title>

<style type="text/css">

@font-face {
font-family: myFirstFont;
src: url(font/Roboto-Regular.ttf);
/*font-weight: bold;*/
}

body
{
```

```
font-family: myFirstFont;
}
.sub_btn
{
background: green;
padding: 10px;
border-radius: 4px;
border: none;
margin-top: 30px;
width: 100%;
color: white;
font-size: 16px;
}
.main_section
{
width: 100%;
```

```
margin: auto;

text-align: center;

}

.form_section

{

width: 50%;

margin: auto;

}

.ticket_table

{

width: 100%;

}

.ticket_table tr

{

line-height: 40px;

font-size: 18px;
```

```
}

.ticket_table td input
{
padding: 7px;
width: 100%;
}
.ticket_table td select
{
width: 100%;
padding: 7px;
}
body
{
background-image: url("img.jpg");
/*height: 100%;*/
```

```
/*background: linear-gradient(rgb(193 196 225 / 80%), rgb(237 158 37 / 80%)), url(img.jpg);*/
background-position: center;
background-repeat: no-repeat;
background-size: cover;
}
.navbar
{
width: 100%;
height: 60px;
}
.navbar_ul
{
float: right;
}
```

```
.navbar_li
{
float: left;
background-color: royalblue;
padding: 10px;
margin-right: 10px;
list-style: none;
}
.nav-icon
{
color: white;
padding: 10px;
text-decoration: none;
}
</style>
```

```
</head>

<body>

<div class="main_section">

<div class="navbar">

<ul class="navbar_ul">

<li class="navbar_li"><a href="./home.html"
class="nav-icon">Home</a></li>

<li class="navbar_li"><a
href="./prediction.html" class="nav-
icon">Predict</a></li>

</ul>

</div>

<h1 >Flight Delay Prediction</h1>


<div class="form_section">

<form action="./flight_delay_result.html">
```

```
<table class="ticket_table">

<tr>

<td>Airline</td>

<td>

<select name="airline">

<option value="">Select</option>

<option value="airindia">Air India</option>

<option value="airasia">Air Asia</option>

</select>

</td>

</tr>

<tr>

<td>Source</td>

<td>

<select name="airline">

<option value="">Select</option>
```

```
<option value="Banglore">Banglore</option>
<option value="Chennai">Chennai</option>
</select>
</td>
</tr>
<tr>
<td>Destination</td>
<td>
<select name="airline">
<option value="">Select</option>
<option value="Banglore">Banglore</option>
<option value="Chennai">Chennai</option>
</select>
</td>
</tr>
<tr>
```

```
<td>Dep Date</td>

<td>

<input type="text" name="dep_date">

</td>

</tr>

<tr>

<td>Dep Month</td>

<td>

<input type="text" name="dep_month">

</td>

</tr>

<tr>

<td>Dep Year</td>

<td>

<input type="text" name="dep_year">

</td>
```

```
</tr>

<tr>

<td>Dep Time in Hour</td>

<td>

<input type="text" name="dep_time_hour">

</td>

</tr>

<tr>

<td>Dep Time in mins</td>

<td>

<input type="text" name="dep_time_mins">

</td>

</tr>

<tr>

<td>Arrival Time</td>

<td>
```

```
<input type="text" name="arr_time">
</td>
</tr>

<tr>
<td>Arrival hour</td>
<td>
<input type="text" name="arr_hour">
</td>
</tr>
<tr>
<td>Arrival time in mins</td>
<td>
<input type="text" name="arr_mins">
</td>
</tr>
```

```
</table>

<button type="submit" class="sub_btn">
Submit </button>

</form>

</div>

</div>


</body>


</html>
```

3.Submit.html

```
<html>

<head>
<title>Flight Delay Prediction</title>

<style type="text/csv">

@font-face {
font-family: myFirstFont;
src: url(font/Roboto-Regular.ttf);
/*font-weight: bold;*/
}

body
```

```
{
font-family: myFirstFont;
}
.main_section
{
width: 100%;
margin: auto;
text-align: center;
}
.form_section
{
width: 50%;
margin: auto;
}

body
```

```
{
background-image: url("img.jpg");
/*height: 100%;*/

/*background: linear-gradient(rgb(193 196 225 /
80%), rgb(237 158 37 / 80%)), url(img.jpg);*/
background-position: center;
background-repeat: no-repeat;
background-size: cover;
}
.navbar
{
width: 100%;
height: 60px;
}
.navbar_ul
```

```
{
float: right;
}
.navbar_li
{
float: left;
background-color: royalblue;
padding: 10px;
margin-right: 10px;
list-style: none;
}
.nav-icon
{
color: white;
padding: 10px;
text-decoration: none;
}
</style>


</head>
<body>
<div class="main_section">
<div class="navbar">
<ul class="navbar_ul">
<li class="navbar_li"><a href="./home.html"
class="nav-icon">Home</a></li>
<li class="navbar_li"><a
href="./prediction.html" class="nav-
icon">Predict</a></li>
</ul>
</div>


<h1 >Flight Delay Prediction</h1>
```

```
<div class="form_section">
<p id="delay_result">
Based on the given input, we can get the flight
delay INR.
</p>
</div>
</div>

</body>

</html>
```

4.App.py

```
from flask import Flask,render_template,request
import numpy as np
import pickle
model=pickle.load(open(r”model1.pkl”,’rb’))
@app.route(“/home)
Def home():
   return render_template(‘home.html’)
@app.route(“/predict”)
Def home():
Return render_template (‘predict.html’)


@app.route(“/pred”,methods=[‘POST’,’GET’])
Def predict():
X=[[int(x) for x in request.form.value()]]
```

```
Print(x)
 X=np.array(x)
Print(x.shape)
 Print(x)
Pred=model.predict(x)
Print(pred)
Return render_template
('submit.html,prediction_text=pred)


If __name__=="__main__":
App.run(debug=False)
```