SPRINT – 1 PROJECT DOCUMENT

Date	12 November 2022
Team ID	PNT2022TMID13084
Project Name	Flight Delay Prediction Using Machine Learning

DEVELOPMENT PHASE:

SPRINT-1:

Outline:

1. Data Pre-processing
2. Data Analysis
3. Model Building
4. Saving Best Model

Required Libraries:

Pandas	- Data Pre-processing
Numpy	- Data Pre-processing, Analysis
Matplotlib	- Visualization
Seaborn	- Visualization
Sklearn	- Model Building
Pickle	- Model saving

Software/Tool:

- Google colab
- Used Language Python

Data Pre-processing:

Data Collection:

Dataset is collected from the IBM career smartinternz portal in Guided Project.

Dataset description:

The dataset contains 26 variables with various data types such as string, object, time, integer, float.

```
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
```

Columns Description:

Dest means Destination Airport.

Crs_dep_time and crs_arr_time is planned departure and arrival time.

Crs_elapsed _time is estimated travel time as per plan.

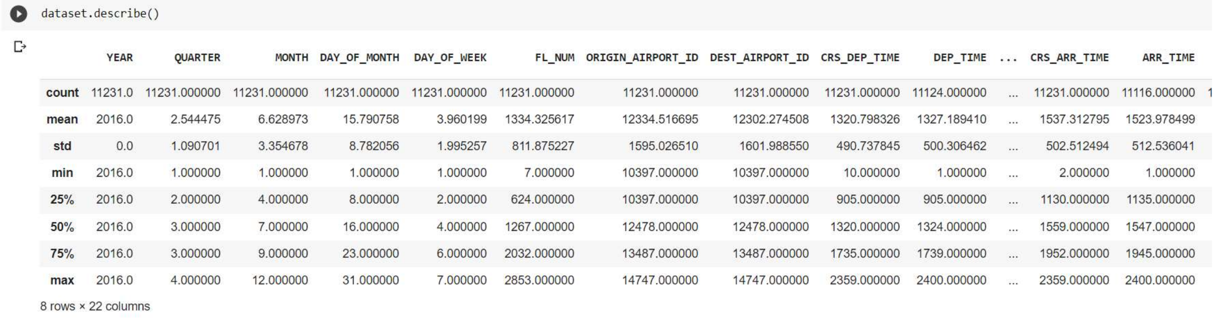
Arr_time and dep_time are actual arrival and departure time.

Actual_elapsed_time is actual travelled time

To pre-process our dataset, we need to import above mentioned required libraries, then import data using pandas.

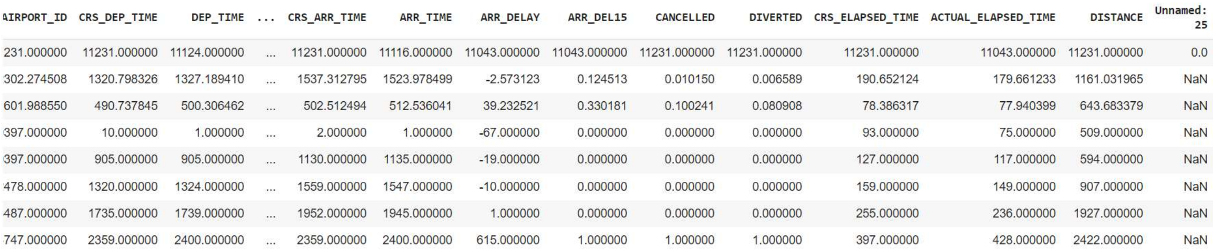
This data does not contain any duplicated values and null values except in arrival , departure time columns, because these left empty when flights are cancelled.

Descriptive Analytics:



```
dataset.describe()
```

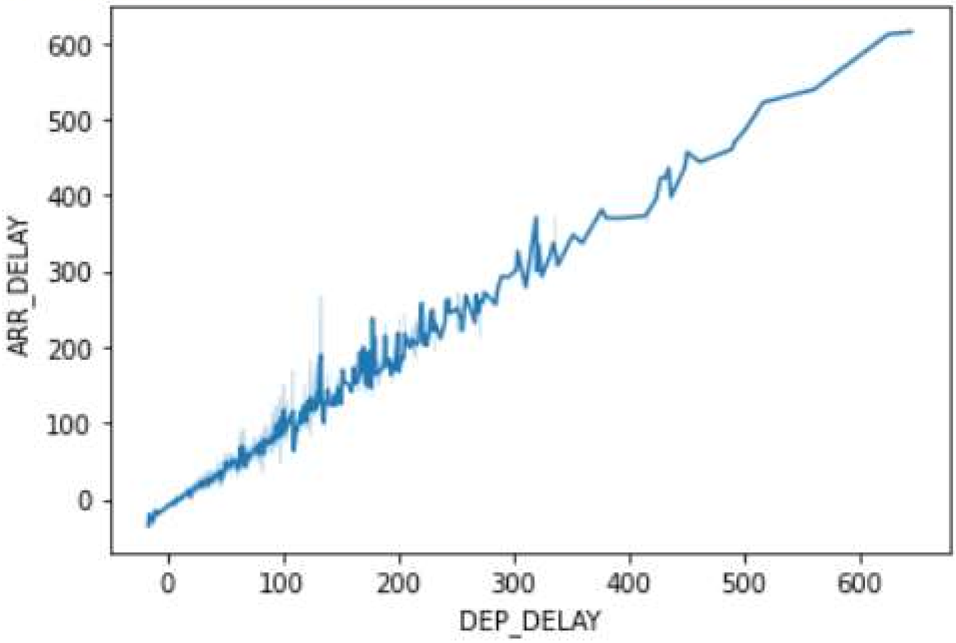
```
         YEAR       QUARTER         MONTH  DAY_OF_MONTH   DAY_OF_WEEK        FL_NUM  ORIGIN_AIRPORT_ID  DEST_AIRPORT_ID  CRS_DEP_TIME      DEP_TIME  ...  CRS_ARR_TIME      ARR_TIME
count 11231.0  11231.000000  11231.000000  11231.000000  11231.000000  11231.000000       11231.000000     11231.000000  11231.000000  11124.000000  ...  11231.000000  11116.000000  1
mean   2016.0      2.544475      6.628973     15.790758      3.960199   1334.325617       12334.516695     12302.274508   1320.798326   1327.189410  ...   1537.312795   1523.978499
std       0.0      1.090701      3.354678      8.782056      1.995257    811.875227        1595.026510      1601.988550    490.737845    500.306462  ...    502.512494    512.536041
min    2016.0      1.000000      1.000000      1.000000      1.000000      7.000000       10397.000000     10397.000000     10.000000      1.000000  ...      2.000000      1.000000
25%    2016.0      2.000000      4.000000      8.000000      2.000000    624.000000       10397.000000     10397.000000    905.000000    905.000000  ...   1130.000000   1135.000000
50%    2016.0      3.000000      7.000000     16.000000      4.000000   1267.000000       12478.000000     12478.000000   1320.000000   1324.000000  ...   1559.000000   1547.000000
75%    2016.0      3.000000      9.000000     23.000000      6.000000   2032.000000       13487.000000     13487.000000   1735.000000   1739.000000  ...   1952.000000   1945.000000
max    2016.0      4.000000     12.000000     31.000000      7.000000   2853.000000       14747.000000     14747.000000   2359.000000   2400.000000  ...   2359.000000   2400.000000
8 rows × 22 columns
```



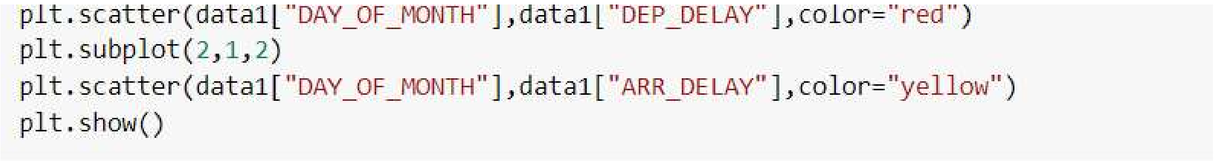
```
AIRPORT_ID  CRS_DEP_TIME      DEP_TIME  ...  CRS_ARR_TIME      ARR_TIME     ARR_DELAY     ARR_DEL15     CANCELLED      DIVERTED  CRS_ELAPSED_TIME  ACTUAL_ELAPSED_TIME      DISTANCE  Unnamed:
                                                                                                                                                                                 25
231.000000  11231.000000  11124.000000  ...  11231.000000  11116.000000  11043.000000  11043.000000  11231.000000  11231.000000      11231.000000         11043.000000  11231.000000       0.0
302.274508   1320.798326   1327.189410  ...   1537.312795   1523.978499     -2.573123      0.124513      0.010150      0.006589        190.652124           179.661233   1161.031965       NaN
601.988550    490.737845    500.306462  ...    502.512494    512.536041     39.232521      0.330181      0.100241      0.080908         78.386317            77.940399    643.683379       NaN
397.000000     10.000000      1.000000  ...      2.000000      1.000000    -67.000000      0.000000      0.000000      0.000000         93.000000            75.000000    509.000000       NaN
397.000000    905.000000    905.000000  ...   1130.000000   1135.000000    -19.000000      0.000000      0.000000      0.000000        127.000000           117.000000    594.000000       NaN
478.000000   1320.000000   1324.000000  ...   1559.000000   1547.000000    -10.000000      0.000000      0.000000      0.000000        159.000000           149.000000    907.000000       NaN
487.000000   1735.000000   1739.000000  ...   1952.000000   1945.000000      1.000000      0.000000      0.000000      0.000000        255.000000           236.000000   1927.000000       NaN
747.000000   2359.000000   2400.000000  ...   2359.000000   2400.000000    615.000000      1.000000      1.000000      1.000000        397.000000           428.000000   2422.000000       NaN
```

Data Analysis And Visualization:

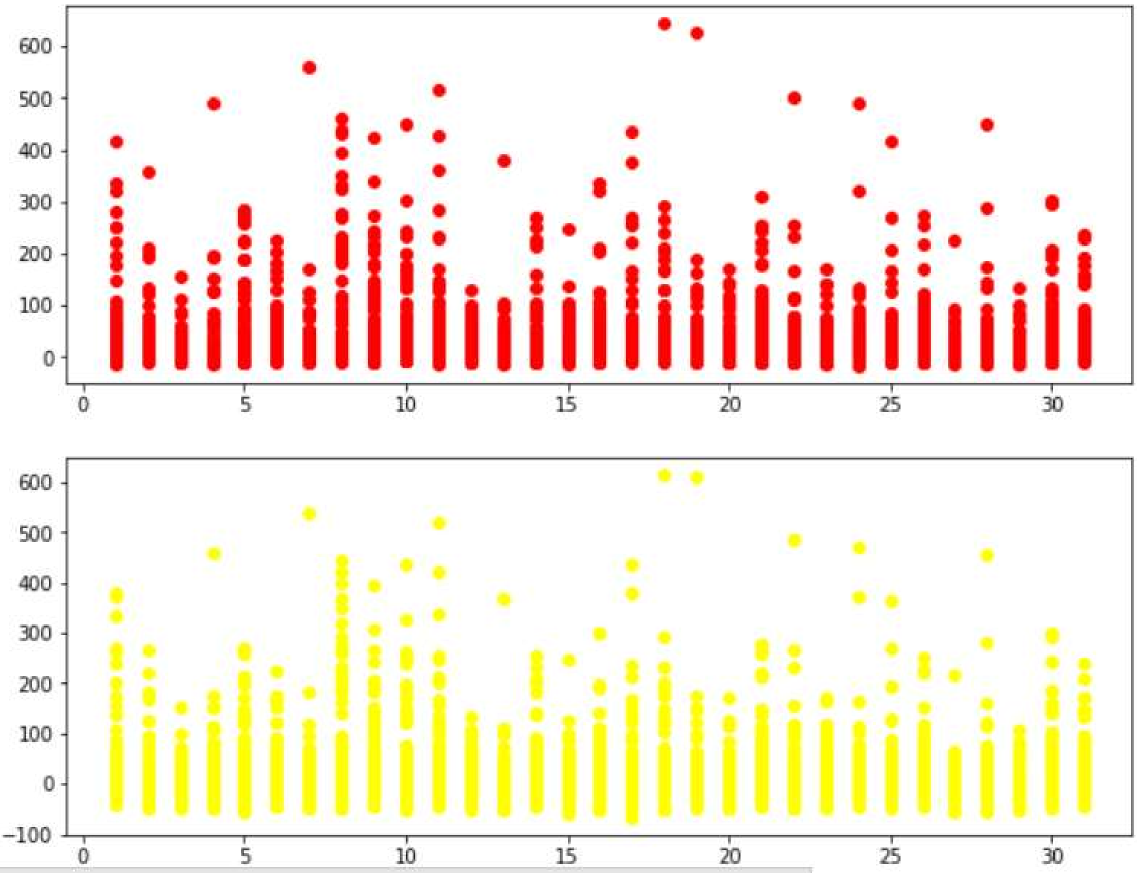
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5f3feb7b50>
```



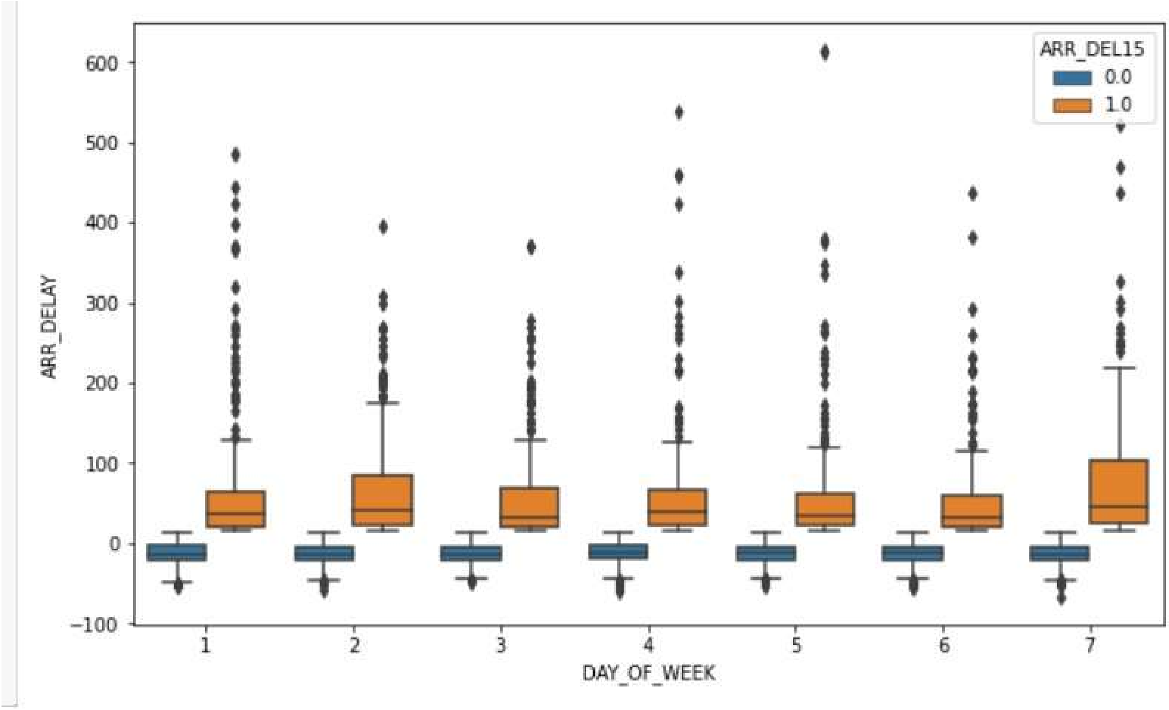
This graph shows the positive trend and strong binding between arrival and departure delay.



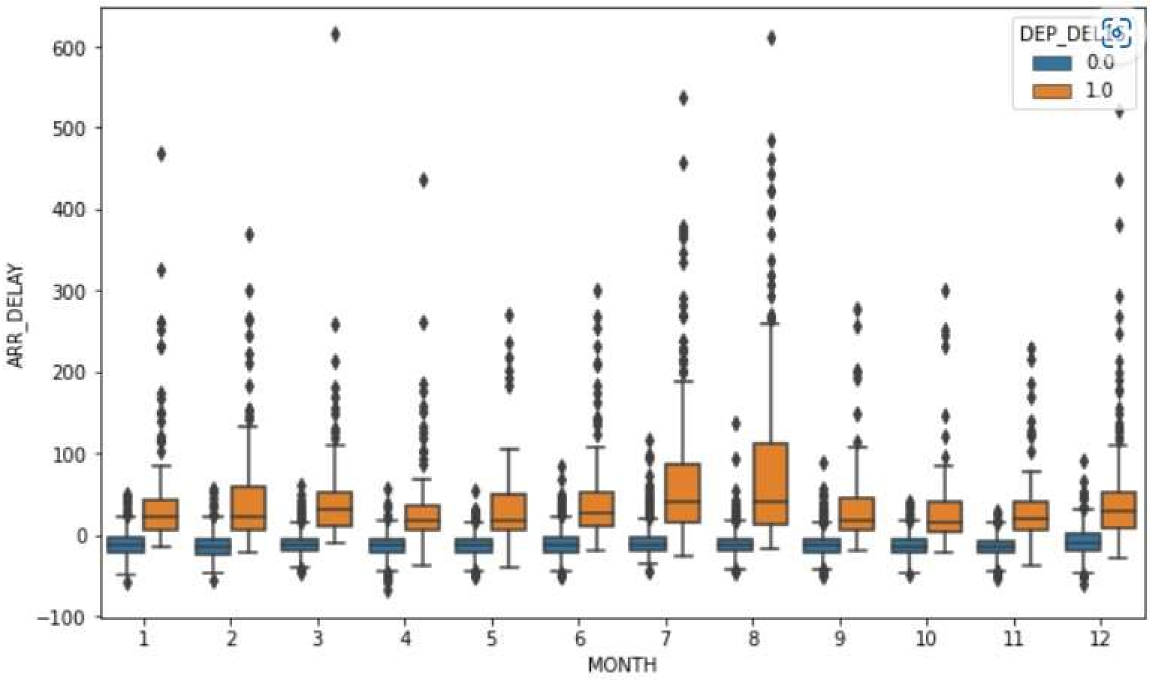
```
plt.scatter(data1["DAY_OF_MONTH"],data1["DEP_DELAY"],color="red")
plt.subplot(2,1,2)
plt.scatter(data1["DAY_OF_MONTH"],data1["ARR_DELAY"],color="yellow")
plt.show()
```



This above picture shows the relationship between day of month and delays.

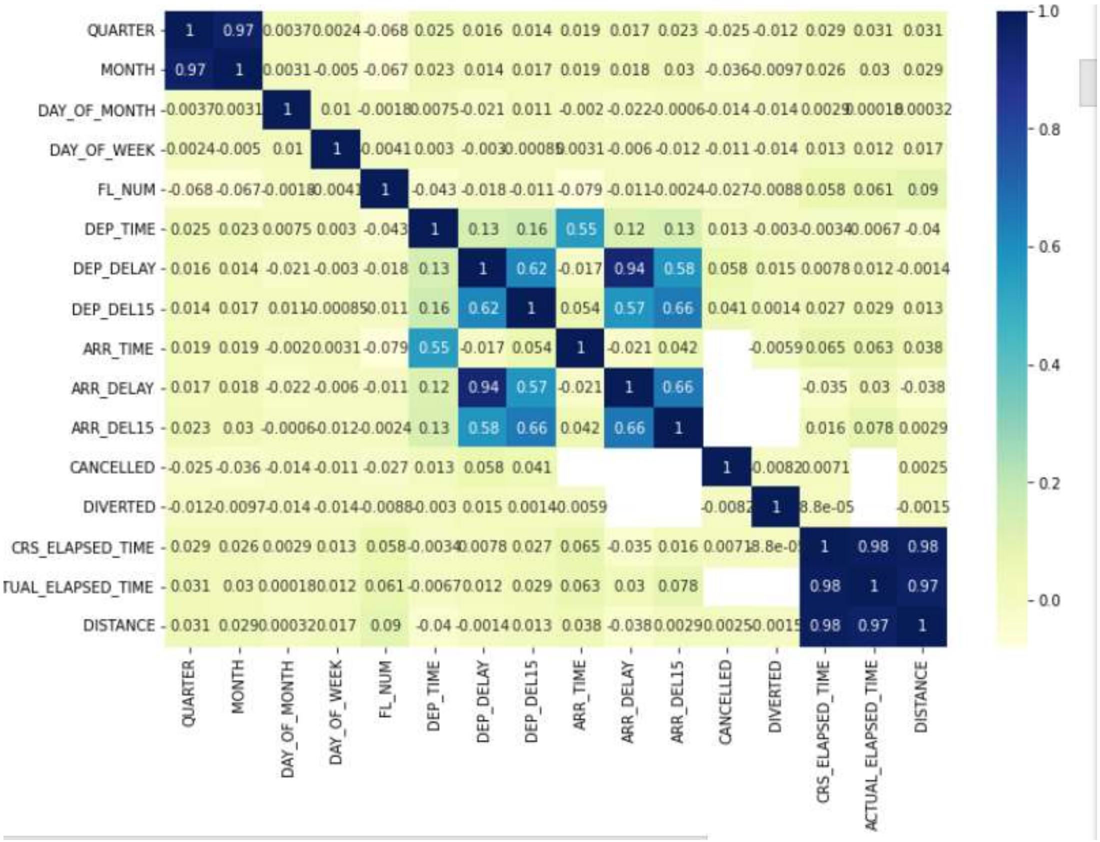


This above boxplot shows the trends of days of the week and delay, Monday and Saturday had high delays.



This above boxplot shows the seasonal relationship between months and delays. August had highest no of delays.

Correlation between columns:



	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_NUM	DEP_TIME	DEP_DELAY	DEP_DEL15	ARR_TIME	ARR_DELAY	ARR_DEL15	CANCELLED	DIVERTED	CRS_ELAPSED_TIME	ACTUAL_ELAPSED_TIME	DISTANCE
QUARTER	1	0.97	-0.0037	-0.0024	-0.068	0.025	0.016	0.014	0.019	0.017	0.023	-0.025	-0.012	0.029	0.031	0.031
MONTH	0.97	1	0.0031	-0.005	-0.067	0.023	0.014	0.017	0.019	0.018	0.03	-0.036	-0.0097	0.026	0.03	0.029
DAY_OF_MONTH	-0.0037	0.0031	1	0.01	-0.0018	0.0075	-0.021	0.011	-0.002	-0.022	-0.0006	-0.014	-0.014	0.0029	0.00018	0.00032
DAY_OF_WEEK	-0.0024	-0.005	0.01	1	0.0041	0.003	-0.003	-0.00085	0.0031	-0.006	-0.012	-0.011	-0.014	0.013	0.012	0.017
FL_NUM	-0.068	-0.067	-0.0018	0.0041	1	-0.043	-0.018	-0.011	-0.079	-0.011	-0.0024	-0.027	-0.0088	0.058	0.061	0.09
DEP_TIME	0.025	0.023	0.0075	0.003	-0.043	1	0.13	0.16	0.55	0.12	0.13	0.013	-0.003	-0.0034	-0.0067	-0.04
DEP_DELAY	0.016	0.014	-0.021	-0.003	-0.018	0.13	1	0.62	-0.017	0.94	0.58	0.058	0.015	0.0078	0.012	-0.0014
DEP_DEL15	0.014	0.017	0.011	-0.00085	-0.011	0.16	0.62	1	0.054	0.57	0.66	0.041	0.0014	0.027	0.029	0.013
ARR_TIME	0.019	0.019	-0.002	0.0031	-0.079	0.55	-0.017	0.054	1	-0.021	0.042		-0.0059	0.065	0.063	0.038
ARR_DELAY	0.017	0.018	-0.022	-0.006	-0.011	0.12	0.94	0.57	-0.021	1	0.66			-0.035	0.03	-0.038
ARR_DEL15	0.023	0.03	-0.0006	-0.012	-0.0024	0.13	0.58	0.66	0.042	0.66	1			0.016	0.078	0.0029
CANCELLED	-0.025	-0.036	-0.014	-0.011	-0.027	0.013	0.058	0.041				1	-0.0082	0.0071		0.0025
DIVERTED	-0.012	-0.0097	-0.014	-0.014	-0.0088	-0.003	0.015	0.0014	-0.0059			-0.0082	1	8.8e-05		-0.0015
CRS_ELAPSED_TIME	0.029	0.026	0.0029	0.013	0.058	-0.0034	0.0078	0.027	0.065	-0.035	0.016	0.0071	8.8e-05	1	0.98	0.98
TUAL_ELAPSED_TIME	0.031	0.03	0.00018	0.012	0.061	-0.0067	0.012	0.029	0.063	0.03	0.078			0.98	1	0.97
DISTANCE	0.031	0.029	0.00032	0.017	0.09	-0.04	-0.0014	0.013	0.038	-0.038	0.0029	0.0025	-0.0015	0.98	0.97	1

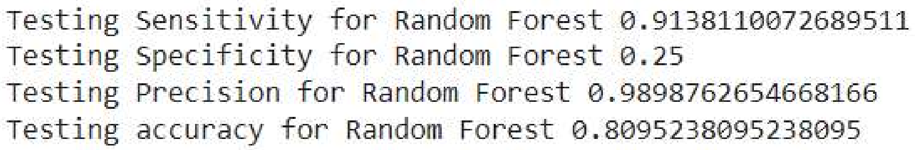
Model Buliding:

We builded

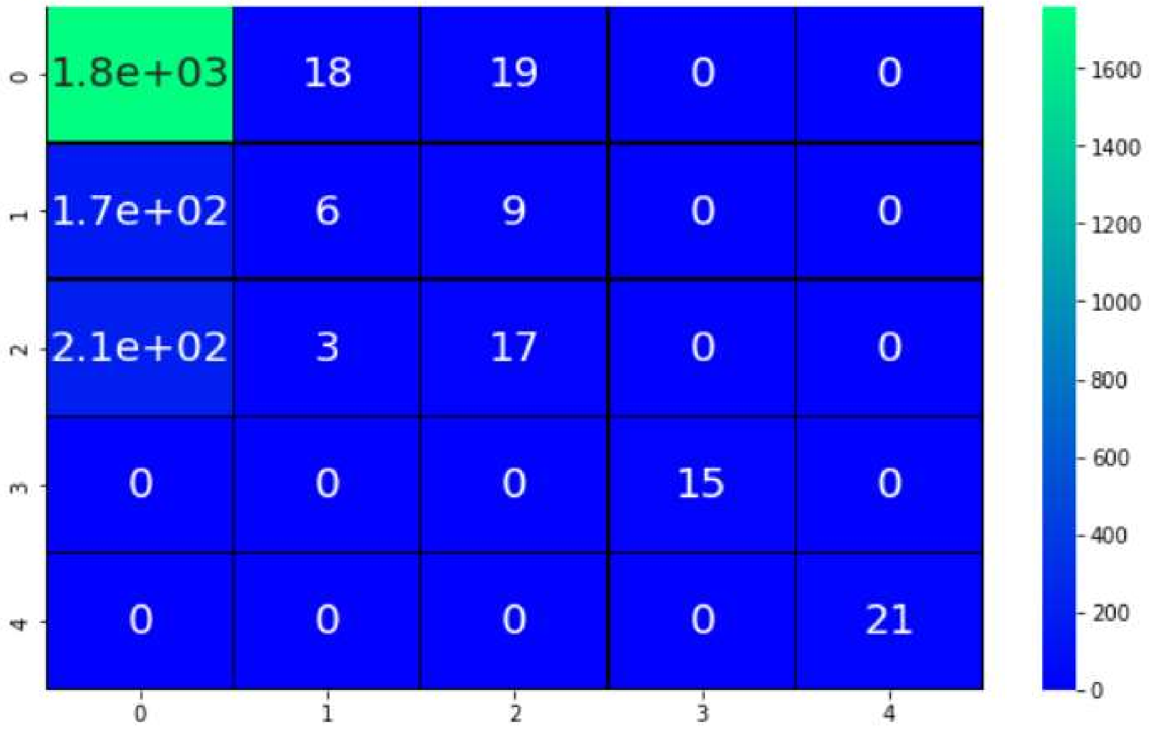
```
Decision Tree with 0.8376550169109357
Random Forest with 0.8095238095238095
SVM with 0.7378727191811304
KNN with 0.7894971072541166
Logistic Regession with 0.7997329773030708
```

We will explore only Random Forest and Decision Tree which have high accuracy

Random Forest:

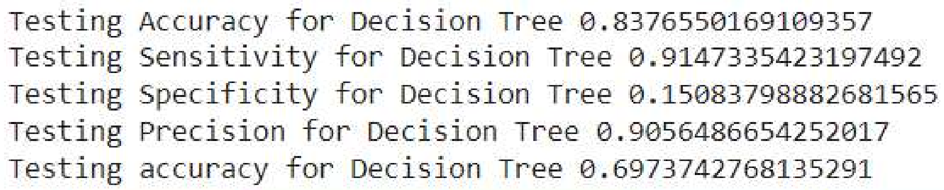


```
Testing Sensitivity for Random Forest 0.9138110072689511
Testing Specificity for Random Forest 0.25
Testing Precision for Random Forest 0.9898762654668166
Testing accuracy for Random Forest 0.8095238095238095
```

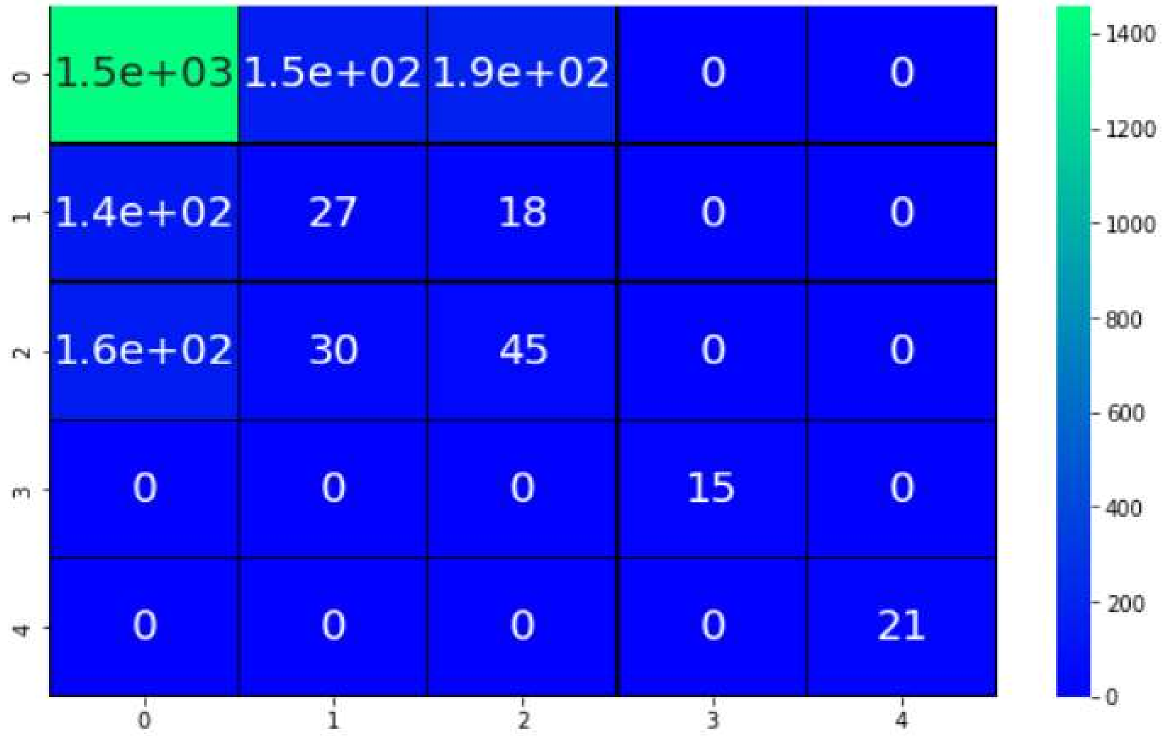


	0	1	2	3	4
0	1.8e+03	18	19	0	0
1	1.7e+02	6	9	0	0
2	2.1e+02	3	17	0	0
3	0	0	0	15	0
4	0	0	0	0	21

Decision Tree:

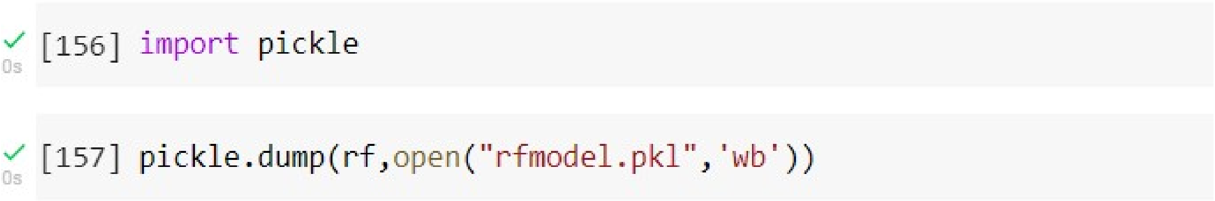


```
Testing Accuracy for Decision Tree 0.8376550169109357
Testing Sensitivity for Decision Tree 0.9147335423197492
Testing Specificity for Decision Tree 0.15083798882681565
Testing Precision for Decision Tree 0.9056486654252017
Testing accuracy for Decision Tree 0.6973742768135291
```



	0	1	2	3	4
0	1.5e+03	1.5e+02	1.9e+02	0	0
1	1.4e+02	27	18	0	0
2	1.6e+02	30	45	0	0
3	0	0	0	15	0
4	0	0	0	0	21

Model Saving: Random Forest gives the best accuracy then others , so we save random forest model using pickle.



```
[156] import pickle
```

```
[157] pickle.dump(rf,open("rfmodel.pkl",'wb'))
```

Conclusion: In this sprint , we builded our model , evaluated and saved. In next sprint, we deploy our model IBM cloud using IBM Watson and building Dashboard.