SPRINT – 1 PROJECT DOCUMENT

Team ID	PNT2022TMID53187
Project Name	Flight Delay Prediction Using Machine Learning

DEVELOPMENT PHASE:

SPRINT-1:

Outline:

1. Data Pre-processing
2. EDA/Data Analysis
3. Feature Engineering
4. Model Building
5. Saving Best Model

Required Libraries:

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

Software/Tool:

- Anaconda- Jupyter Notebook
- Used Language Python

Data Pre-processing:

Data Collection:

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

Dataset description:

In [7]:
```
dataset.describe()
```

Out[7]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_NUM	ORIGIN_AIRPORT_ID	DEST_AIRPORT_ID	CRS_DEP_TIME	DEP_TIME	...	CRS_ARR_T
count	11231.0	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11124.000000	...	11231.00
mean	2016.0	2.544475	6.628973	15.790758	3.960199	1334.325617	12334.516695	12302.274508	1320.798326	1327.189410	...	1537.31
std	0.0	1.090701	3.354678	8.782056	1.995257	811.875227	1595.026510	1601.988550	490.737845	500.306462	...	502.51
min	2016.0	1.000000	1.000000	1.000000	1.000000	7.000000	10397.000000	10397.000000	10.000000	1.000000	...	2.00
25%	2016.0	2.000000	4.000000	8.000000	2.000000	624.000000	10397.000000	10397.000000	905.000000	905.000000	...	1130.00
50%	2016.0	3.000000	7.000000	16.000000	4.000000	1267.000000	12478.000000	12478.000000	1320.000000	1324.000000	...	1559.00
75%	2016.0	3.000000	9.000000	23.000000	6.000000	2032.000000	13487.000000	13487.000000	1735.000000	1739.000000	...	1952.00
max	2016.0	4.000000	12.000000	31.000000	7.000000	2853.000000	14747.000000	14747.000000	2359.000000	2400.000000	...	2359.00

8 rows × 22 columns

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:

In [7]:
```python
dataset.describe()
```

Out[7]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_NUM	ORIGIN_AIRPORT_ID	DEST_AIRPORT_ID	CRS_DEP_TIME	DEP_TIME	...	CRS_ARR_T
count	11231.0	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11124.000000	...	11231.00
mean	2016.0	2.544475	6.628973	15.790758	3.960199	1334.325617	12334.516695	12302.274508	1320.798326	1327.189410	...	1537.31
std	0.0	1.090701	3.354678	8.782056	1.995257	811.875227	1595.026510	1601.988550	490.737845	500.306462	...	502.51
min	2016.0	1.000000	1.000000	1.000000	1.000000	7.000000	10397.000000	10397.000000	10.000000	1.000000	...	2.00
25%	2016.0	2.000000	4.000000	8.000000	2.000000	624.000000	10397.000000	10397.000000	905.000000	905.000000	...	1130.00
50%	2016.0	3.000000	7.000000	16.000000	4.000000	1267.000000	12478.000000	12478.000000	1320.000000	1324.000000	...	1559.00
75%	2016.0	3.000000	9.000000	23.000000	6.000000	2032.000000	13487.000000	13487.000000	1735.000000	1739.000000	...	1952.00
max	2016.0	4.000000	12.000000	31.000000	7.000000	2853.000000	14747.000000	14747.000000	2359.000000	2400.000000	...	2359.00

8 rows × 22 columns

In [7]:
```python
dataset.describe()
```

Out[7]:

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

Data cleaning and analysis:

In [8]:
```python
dataset.isnull().sum()
```

Out[8]:
```
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
Unnamed: 25           11231
dtype: int64
```

In [9]:
```python
dataset['DEST'].unique()
```

Out[9]:
```
array(['SEA', 'MSP', 'DTW', 'ATL', 'JFK'], dtype=object)
```

In [12]:
```
dataset=dataset[["FL_NUM","MONTH","DAY_OF_MONTH","DAY_OF_WEEK","ORIGIN","DEST","CRS_ARR_TIME","DEP_DEL15","ARR_DEL15"]]
dataset.isnull().sum()
```

Out[12]:
```
FL_NUM            0
MONTH             0
DAY_OF_MONTH      0
DAY_OF_WEEK       0
ORIGIN            0
DEST              0
CRS_ARR_TIME      0
DEP_DEL15       107
ARR_DEL15       188
dtype: int64
```

In [13]:
```
dataset = dataset.fillna({'ARR_DEL15':1})
dataset =dataset.fillna({'DEP_DEL15':0})
dataset.iloc[177:185]
```

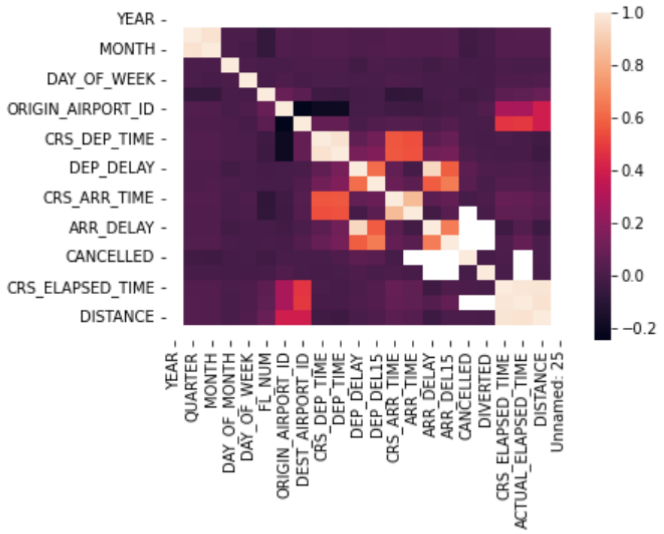
Out[13]:

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
177	2834	1	9	6	MSP	SEA	852	0.0	1.0
178	2839	1	9	6	DTW	JFK	1724	0.0	0.0
179	86	1	10	7	MSP	DTW	1632	0.0	1.0
180	87	1	10	7	DTW	MSP	1649	1.0	0.0
181	423	1	10	7	JFK	ATL	1600	0.0	0.0
182	440	1	10	7	JFK	ATL	849	0.0	0.0
183	485	1	10	7	JFK	SEA	1945	1.0	0.0
184	557	1	10	7	MSP	DTW	912	0.0	1.0

Heatmap and data correlation:

In [10]:
```
sns.heatmap(dataset.corr())
```

Out[10]:



Feature Engineering:

In [16]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
dataset['DEST']=le.fit_transform(dataset['DEST'])
dataset['ORIGIN']=le.fit_transform(dataset['ORIGIN'])
dataset.head(5)
```

Out[16]:

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
0	1399	1	1	5	0	4	2143	0.0	0.0
1	1476	1	1	5	1	3	1435	0.0	0.0
2	1597	1	1	5	0	4	1215	0.0	0.0
3	1768	1	1	5	4	3	1335	0.0	0.0
4	1823	1	1	5	4	1	607	0.0	0.0

In [17]:

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

Out[17]:

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15	ORIGIN_0	ORIGIN_1	ORIGIN_2	ORIGIN_3	ORIGIN_4	DEST_0	DEST_1	DEST_2
0	1399	1	1	5	2143	0.0	0.0	1	0	0	0	0	0	0	C
1	1476	1	1	5	1435	0.0	0.0	0	1	0	0	0	0	0	C
2	1597	1	1	5	1215	0.0	0.0	1	0	0	0	0	0	0	C
3	1768	1	1	5	1335	0.0	0.0	0	0	0	0	1	0	0	C
4	1823	1	1	5	607	0.0	0.0	0	0	0	0	1	0	1	C

One-hot encoding and Model Training:

```
In [19]: from sklearn.preprocessing import OneHotEncoder
         oh=OneHotEncoder()
         z=oh.fit_transform(x[:,4:5]).toarray()
         t=oh.fit_transform(x[:,5:6]).toarray()

In [20]: z

Out[20]: 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.]])

In [21]: t

Out[21]: array([[1., 0.],
                [1., 0.],
                [1., 0.],
                ...,
                [1., 0.],
                [1., 0.],
                [1., 0.]])

In [23]: 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)
         x_test.shape

Out[23]: (2247, 8)
```

```
In [24]: x_train.shape

Out[24]: (8984, 8)

In [25]: y_test.shape

Out[25]: (2247, 1)

In [26]: y_train.shape

Out[26]: (8984, 1)
```

Decision tree:

```
In [27]: from sklearn.tree import DecisionTreeClassifier
         clf = DecisionTreeClassifier(max_depth = 4, min_samples_split = 4, random_state = 0)

In [28]: clf.fit(x_train, y_train)

Out[28]: DecisionTreeClassifier(max_depth=4, min_samples_split=4, random_state=0)

In [29]: pred = clf.predict(x_test)

In [31]: decisiontree = clf.predict(x_test)
         decisiontree

Out[31]: array([1, 0, 0, ..., 0, 0, 0], dtype=uint8)

In [32]: from sklearn.metrics import accuracy_score
         print(accuracy_score(y_test, decisiontree))

         0.8255451713395638
```

Model Saving:

```
In [71]: import pickle
```

```
In [72]: 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.