```
import sys
import numpy as np
import pandas as pd
import seaborn as sns
import pickle
%matplotlib inline
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import sklearn.metrics as metrics
```

```
df=pd.read_csv('C:\\Users\\ketziyal\\Downloads\\flightdata.csv')
```

```
#Analyze the data
```

```
df.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
```

```
df.describe()
```

```
          YEAR       QUARTER         MONTH  DAY_OF_MONTH   DAY_OF_WEEK
\
count  11231.0  11231.000000  11231.000000  11231.000000  11231.000000

mean    2016.0      2.544475      6.628973     15.790758      3.960199

std        0.0      1.090701      3.354678      8.782056      1.995257

min     2016.0      1.000000      1.000000      1.000000      1.000000

25%     2016.0      2.000000      4.000000      8.000000      2.000000

50%     2016.0      3.000000      7.000000     16.000000      4.000000

75%     2016.0      3.000000      9.000000     23.000000      6.000000

max     2016.0      4.000000     12.000000     31.000000      7.000000


             FL_NUM  ORIGIN_AIRPORT_ID  DEST_AIRPORT_ID  CRS_DEP_TIME
\
count  11231.000000       11231.000000     11231.000000  11231.000000

mean    1334.325617       12334.516695     12302.274508   1320.798326

std      811.875227        1595.026510      1601.988550    490.737845

min        7.000000       10397.000000     10397.000000     10.000000

25%      624.000000       10397.000000     10397.000000    905.000000

50%     1267.000000       12478.000000     12478.000000   1320.000000

75%     2032.000000       13487.000000     13487.000000   1735.000000

max     2853.000000       14747.000000     14747.000000   2359.000000


           DEP_TIME  ...  CRS_ARR_TIME      ARR_TIME     ARR_DELAY  \
count  11124.000000  ...  11231.000000  11116.000000  11043.000000
mean    1327.189410  ...   1537.312795   1523.978499     -2.573123
std      500.306462  ...    502.512494    512.536041     39.232521
min        1.000000  ...      2.000000      1.000000    -67.000000
25%      905.000000  ...   1130.000000   1135.000000    -19.000000
```

```
50%     1324.000000  ...   1559.000000   1547.000000    -10.000000
75%     1739.000000  ...   1952.000000   1945.000000      1.000000
max     2400.000000  ...   2359.000000   2400.000000    615.000000

          ARR_DEL15     CANCELLED      DIVERTED  CRS_ELAPSED_TIME  \
count  11043.000000  11231.000000  11231.000000      11231.000000  
mean       0.124513      0.010150      0.006589        190.652124  
std        0.330181      0.100241      0.080908         78.386317  
min        0.000000      0.000000      0.000000         93.000000  
25%        0.000000      0.000000      0.000000        127.000000  
50%        0.000000      0.000000      0.000000        159.000000  
75%        0.000000      0.000000      0.000000        255.000000  
max        1.000000      1.000000      1.000000        397.000000  

       ACTUAL_ELAPSED_TIME      DISTANCE  Unnamed: 25
count         11043.000000  11231.000000          0.0
mean            179.661233   1161.031965          NaN
std              77.940399    643.683379          NaN
min              75.000000    509.000000          NaN
25%             117.000000    594.000000          NaN
50%             149.000000    907.000000          NaN
75%             236.000000   1927.000000          NaN
max             428.000000   2422.000000          NaN

[8 rows x 22 columns]
```

```
#handling missing values
```

```
df.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
Unnamed: 25            11231
dtype: int64
```

```
df['DEST'].unique()
```

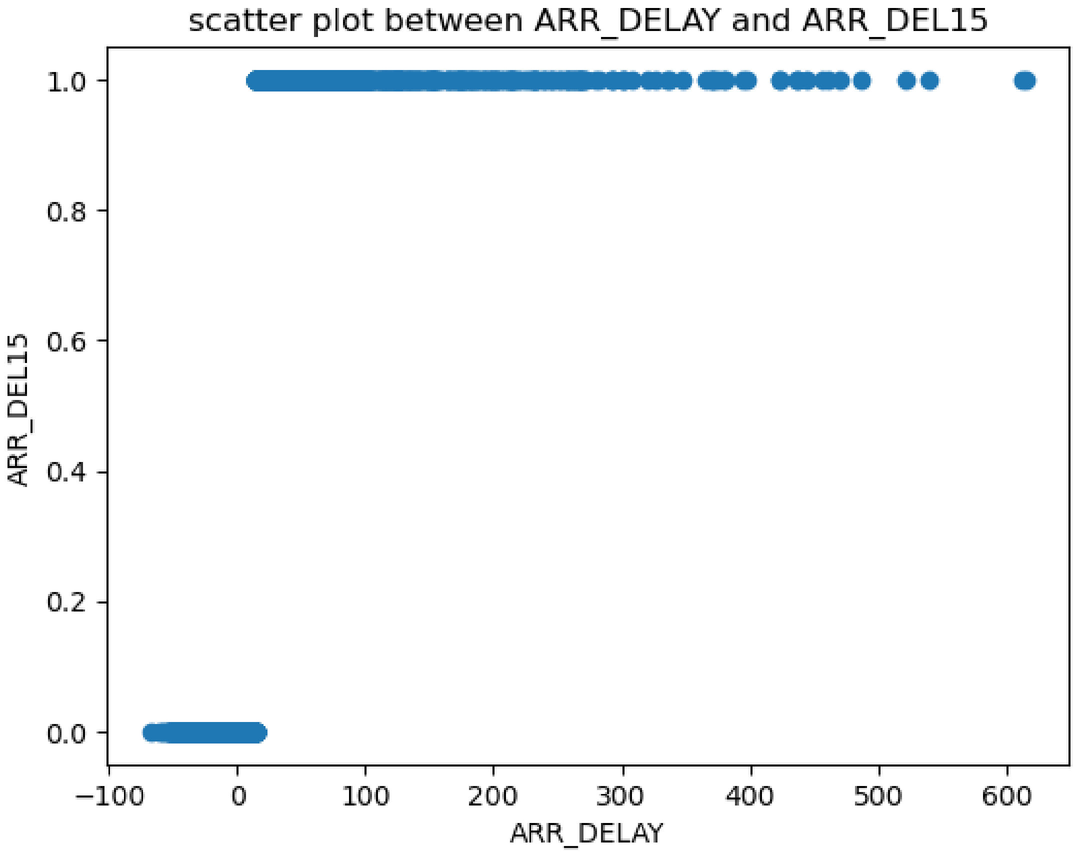
```
array(['SEA', 'MSP', 'DTW', 'ATL', 'JFK'], dtype=object)
```

```
#data visualization
```

```
from matplotlib import pyplot as plt
```

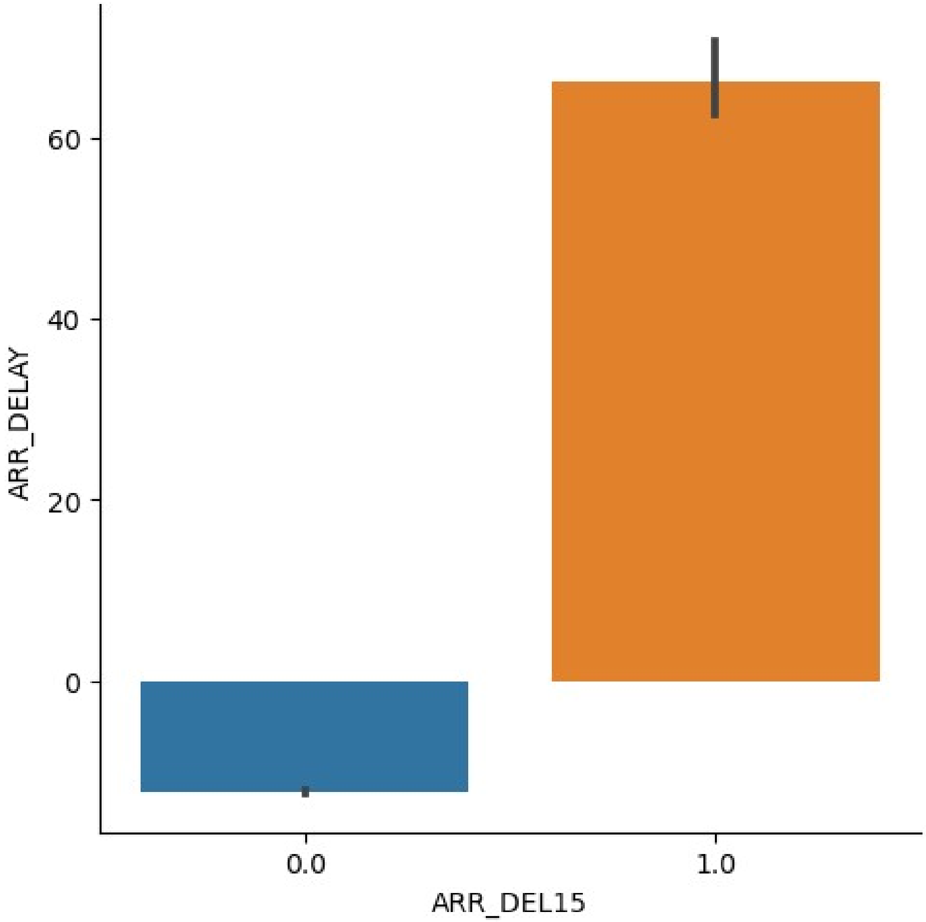
```
plt.scatter(df['ARR_DELAY'],df['ARR_DEL15'])
plt.xlabel('ARR_DELAY')
plt.ylabel('ARR_DEL15')
plt.title('scatter plot between ARR_DELAY and ARR_DEL15')
```

```
Text(0.5, 1.0, 'scatter plot between ARR_DELAY and ARR_DEL15')
```



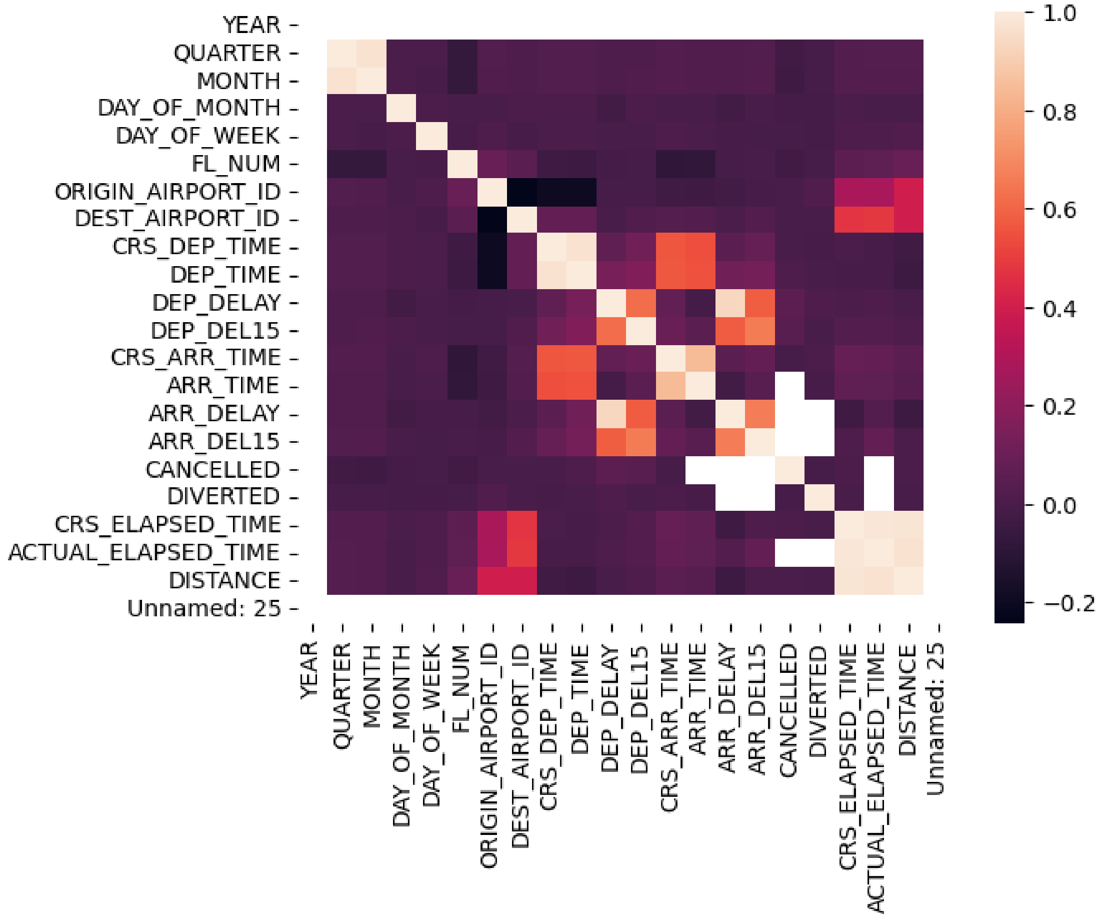
```
sns.catplot(x="ARR_DEL15",y="ARR_DELAY",kind='bar',data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x196d4d7be50>
```



```
sns.heatmap(df.corr())
```

```
<AxesSubplot:>
```



```
df=df.drop('Unnamed: 25',axis=1)
df.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
```

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

```
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
```

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

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

     DEP_DEL15  ARR_DEL15
177        0.0        1.0
178        0.0        0.0
179        0.0        1.0
180        1.0        0.0
```

```
181        0.0        0.0
182        0.0        0.0
183        1.0        0.0
184        0.0        1.0
```

```
import math

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

```
   FL_NUM  MONTH  DAY_OF_MONTH  DAY_OF_WEEK ORIGIN DEST  CRS_ARR_TIME
\
0    1399      1             1            5    ATL  SEA            21

1    1476      1             1            5    DTW  MSP            14

2    1597      1             1            5    ATL  SEA            12

3    1768      1             1            5    SEA  MSP            13

4    1823      1             1            5    SEA  DTW             6


   DEP_DEL15  ARR_DEL15
0        0.0        0.0
1        0.0        0.0
2        0.0        0.0
3        0.0        0.0
4        0.0        0.0
```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

df['DEST']=le.fit_transform(df['DEST'])
df['ORIGIN']=le.fit_transform(df['ORIGIN'])

df.head()
```

```
   FL_NUM  MONTH  DAY_OF_MONTH  DAY_OF_WEEK  ORIGIN  DEST
CRS_ARR_TIME  \
0    1399      1             1            5       0     4
21
1    1476      1             1            5       1     3
14
2    1597      1             1            5       0     4
12
3    1768      1             1            5       4     3
13
4    1823      1             1            5       4     1
6
```

```
   DEP_DEL15  ARR_DEL15
0        0.0        0.0
1        0.0        0.0
2        0.0        0.0
3        0.0        0.0
4        0.0        0.0
```

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

```
t
```

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

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

```
   FL_NUM  MONTH  DAY_OF_MONTH  DAY_OF_WEEK  CRS_ARR_TIME
DEP_DEL15  \
0    1399      1             1            5            21          0.0

1    1476      1             1            5            14          0.0

2    1597      1             1            5            12          0.0

3    1768      1             1            5            13          0.0

4    1823      1             1            5             6          0.0
```

```
   ARR_DEL15  ORIGIN_0  ORIGIN_1  ORIGIN_2  ORIGIN_3  ORIGIN_4  DEST_0  
\
0        0.0         1         0         0         0         0       0   

1        0.0         0         1         0         0         0       0   

2        0.0         1         0         0         0         0       0   

3        0.0         0         0         0         0         1       0   

4        0.0         0         0         0         0         1       0   


   DEST_1  DEST_2  DEST_3  DEST_4  
0       0       0       0       1  
1       0       0       1       0  
2       0       0       0       1  
3       0       0       1       0  
4       1       0       0       0  
```

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

```
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,rando
m_state=0)
```

```
x_test.shape
```

```
(2247, 8)
```

```
x_train.shape
```

```
(8984, 8)
```

```
y_test.shape
```

```
(2247, 1)
```

```
y_train.shape
```

```
(8984, 1)
```

MODEL BUILDING

```
#decision tree
```

```
from sklearn.tree import DecisionTreeClassifier
dc=DecisionTreeClassifier()
dc.fit(x_train,y_train)
dc.score(x_test,y_test)
```

```
0.8607031597685804
```

```
#random forest
```

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=50,random_state=42)
rf.fit(x_train,y_train)
rf.score(x_test,y_test)
```

```
C:\Users\ketziyal\AppData\Local\Temp\ipykernel_19504\905497165.py:3:
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().
  rf.fit(x_train,y_train)
```

```
0.910547396528705
```

```
pd.DataFrame(rf.predict(x_test)).value_counts()
```

```
0.0    2003
1.0     244
dtype: int64
```

```
#logestic regression
```

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(solver='sag')
lr.fit(x_train,y_train)
lr.score(x_test,y_test)
```

```
c:\Users\ketziyal\anaconda3\lib\site-packages\sklearn\utils\
validation.py:1111: 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().
  y = column_or_1d(y, warn=True)
c:\Users\ketziyal\anaconda3\lib\site-packages\sklearn\linear_model\
_sag.py:350: ConvergenceWarning: The max_iter was reached which means
the coef_ did not converge
  warnings.warn(
```

```
0.8615932354250111
```

```
lr.predict(x_test).sum()
```

```
0.0
```

```
#svm
```

```
from sklearn.svm import SVC
svm=SVC(kernel='sigmoid')
svm.fit(x_train,y_train)
svm.score(x_test,y_test)
```

```
c:\Users\ketziyal\anaconda3\lib\site-packages\sklearn\utils\
validation.py:1111: 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().
  y = column_or_1d(y, warn=True)
```

```
0.7725856697819314
```

```
pd.DataFrame(svm.predict(x_test)).value_counts()
```

```
0.0    1941
1.0     306
dtype: int64
```

```
pd.DataFrame(y_test).value_counts()
```

```
0.0    1936
1.0     311
dtype: int64
```

```
# K-NEAREST NEIGHBOUR CLASSIFIER
```

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train,y_train)
knn.score(x_test,y_test)
```

```
c:\Users\ketziyal\anaconda3\lib\site-packages\sklearn\neighbors\
_classification.py:207: 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().
  return self._fit(X, y)
```

```
0.8486871384067646
```

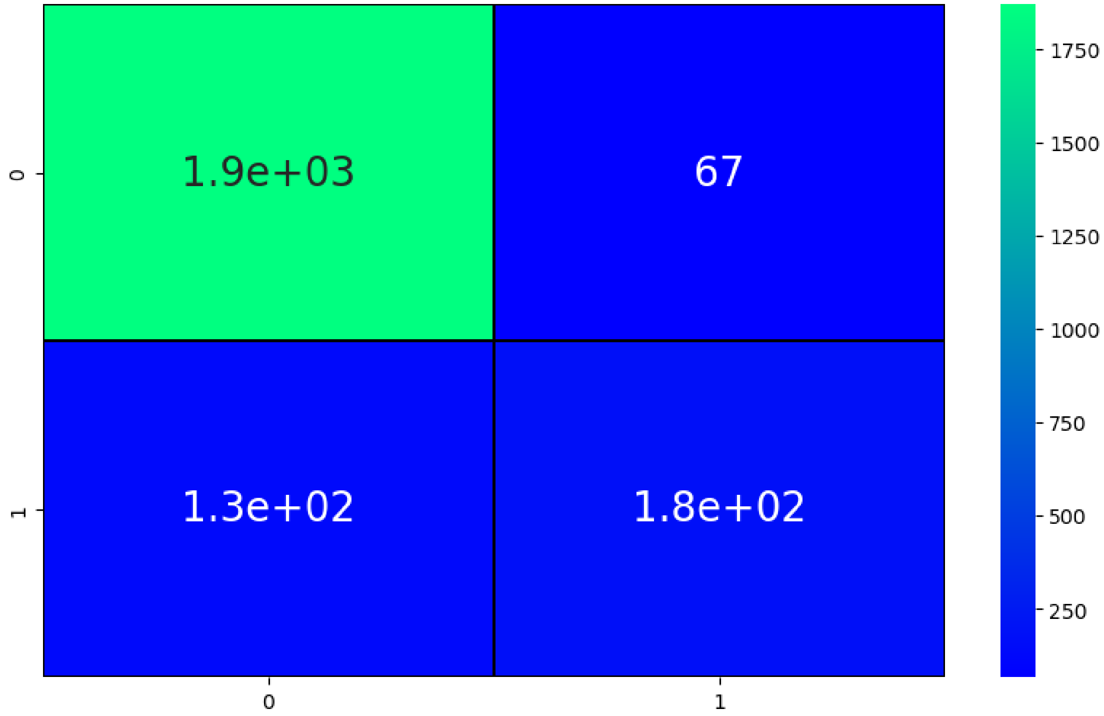
pd.DataFrame(knn.predict(x_test)).value_counts()

Evaluation Of Random Forest

```
from sklearn.metrics import
confusion_matrix,accuracy_score,classification_report
pred=rf.predict(x_test)
cm=confusion_matrix(y_test, pred)
plt.figure(figsize=(10,6))
sns.heatmap(cm, annot=True,cmap='winter',linewidths=0.3,
linecolor='black',annot_kws={"size": 20})
TP=cm[0][0]
TN=cm[1][1]
FN=cm[1][0]
FP=cm[0][1]
#print(round(accuracy_score(prediction3,y_test)*100,2))
#print('Testing Accuracy for knn',(TP+TN)/(TP+TN+FN+FP))
```

```
print('Testing Sensitivity for Random Forest',(TP/(TP+FN)))
print('Testing Specificity for Random Forest',(TN/(TN+FP)))
print('Testing Precision for Random Forest',(TP/(TP+FP)))
print('Testing accuracy for Random Forest',accuracy_score(y_test,
pred))
```

```
Testing Sensitivity for Random Forest 0.9331003494757864
Testing Specificity for Random Forest 0.7254098360655737
Testing Precision for Random Forest 0.9653925619834711
Testing accuracy for Random Forest 0.910547396528705
```

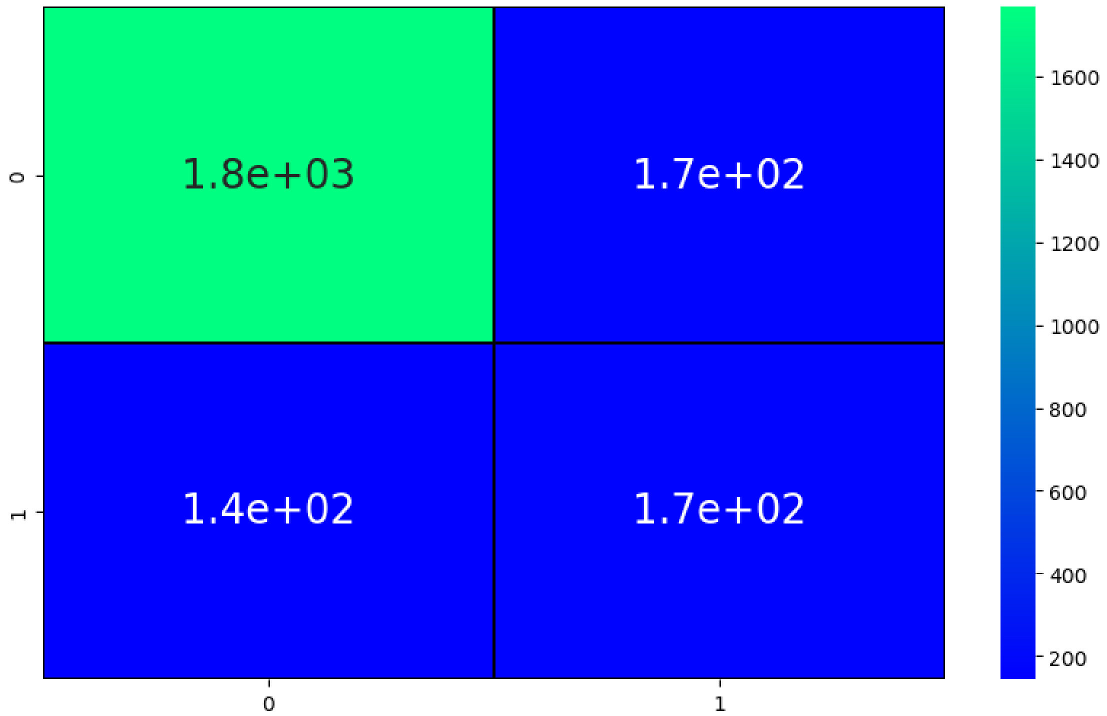


```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

         0.0       0.93      0.97      0.95      1936
         1.0       0.73      0.57      0.64       311

    accuracy                           0.91      2247
   macro avg       0.83      0.77      0.79      2247
weighted avg       0.90      0.91      0.91      2247
```

```
#Evaluation Of Decission Tree

pred1=dc.predict(x_test)
cm1=confusion_matrix(y_test, pred1)
plt.figure(figsize=(10,6))
sns.heatmap(cm1, annot=True,cmap='winter',linewidths=0.3,
linecolor='black',annot_kws={"size": 20})
TP=cm1[0][0]
TN=cm1[1][1]
FN=cm1[1][0]
FP=cm1[0][1]
#print(round(accuracy_score(prediction3,y_test)*100,2))
print('Testing Accuracy for Decision Tree',(TP+TN)/(TP+TN+FN+FP))
print('Testing Sensitivity for Decision Tree',(TP/(TP+FN)))
print('Testing Specificity for Decision Tree',(TN/(TN+FP)))
print('Testing Precision for Decision Tree',(TP/(TP+FP)))
print('Testing accuracy for Decision Tree',accuracy_score(y_test,
pred1))
```

```
Testing Accuracy for Decision Tree 0.8607031597685804
Testing Sensitivity for Decision Tree 0.9246467817896389
Testing Specificity for Decision Tree 0.49702380952380953
Testing Precision for Decision Tree 0.9127066115702479
Testing accuracy for Decision Tree 0.8607031597685804
```



```
print(classification_report(y_test,pred1))
```

```
              precision    recall  f1-score   support

         0.0       0.92      0.91      0.92      1936
         1.0       0.50      0.54      0.52       311

    accuracy                           0.86      2247
   macro avg       0.71      0.72      0.72      2247
```

```
weighted avg       0.87      0.86      0.86      2247
```

```
import pickle

pickle.dump(rf,open("flight.pkl",'wb'))
```