In [1]:
```
# Importing all the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import time
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import precision_score, recall_score, confusion_matrix
```

In [2]:
```
#Setting the values to fit the rows and the columns

start_time = time.time()
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
```

In [3]:
```
#Ignore Warnings

%matplotlib inline
warnings.filterwarnings('ignore')
sns.set_style("darkgrid")
```

In [4]:
```
#Loading the dataset

df = pd.read_csv("flight_status_in_2019 2.csv",index_col= None)
```

In [5]:
```
#Checking the first five records of the dataset

df.head()
```

Out[5]:

	YEAR	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_DATE	OP_UNIQUE_CARRIER	TAIL_NUM	(
0	2019	1	16	3	2019-01-16	AA	N150UW	
1	2019	1	17	4	2019-01-17	AA	N563UW	
2	2019	1	18	5	2019-01-18	AA	N921US	
3	2019	1	19	6	2019-01-19	AA	N604AW	
4	2019	1	20	7	2019-01-20	AA	N975UY	

In [6]:
```
#Checking number of rows and columns

df.shape
```

Out[6]:
```
(8091684, 30)
```

In [10]:
```
#pip install numpy==1.16.5 --upgrade --force-reinstall --user
```

```
Collecting numpy==1.16.5
  Using cached https://files.pythonhosted.org/packages/98/5b/e1bf225ed4614b6a482ea783f75ce571b0d440ba247f6f52c0b7347d6e18/numpy-1.16.5-cp37-cp37m-manylinux1_x86_64.whl (https://files.pythonhosted.org/packages/98/5b/e1bf225ed4614b6a482ea783f75ce571b0d440ba247f6f52c0b7347d6e18/numpy-1.16.5-cp37-cp37m-manylinux1_x86_64.whl)
Installing collected packages: numpy
  Found existing installation: numpy 1.16.5
    Uninstalling numpy-1.16.5:
      Successfully uninstalled numpy-1.16.5
Successfully installed numpy-1.16.5
Note: you may need to restart the kernel to use updated packages.
```

In [7]:
```
#Checking for duplicates

df.drop_duplicates()
df.shape
```

Out[7]:
```
(8091684, 30)
```

In [7]:
```
#Checking the data types

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8091684 entries, 0 to 8091683
Data columns (total 30 columns):
YEAR                    int64
MONTH                   int64
DAY_OF_MONTH            int64
DAY_OF_WEEK             int64
FL_DATE                 object
OP_UNIQUE_CARRIER       object
TAIL_NUM                object
OP_CARRIER_FL_NUM       int64
ORIGIN                  object
DEST                    object
CRS_DEP_TIME            int64
DEP_TIME                float64
DEP_DELAY               float64
CRS_ARR_TIME            int64
ARR_TIME                float64
ARR_DELAY               float64
CANCELLED               float64
CANCELLATION_CODE       object
DIVERTED                float64
CRS_ELAPSED_TIME        float64
ACTUAL_ELAPSED_TIME     float64
AIR_TIME                float64
FLIGHTS                 float64
DISTANCE                float64
CARRIER_DELAY           float64
WEATHER_DELAY           float64
NAS_DELAY               float64
SECURITY_DELAY          float64
LATE_AIRCRAFT_DELAY     float64
Unnamed: 29             float64
dtypes: float64(17), int64(7), object(6)
memory usage: 1.8+ GB
```

In [8]:
```
#Checking the number of missing values

df.isnull().sum()
```

Out[8]:
```
YEAR                      0
MONTH                     0
DAY_OF_MONTH              0
DAY_OF_WEEK               0
FL_DATE                   0
OP_UNIQUE_CARRIER         0
TAIL_NUM                  28514
OP_CARRIER_FL_NUM         0
ORIGIN                    0
DEST                      0
CRS_DEP_TIME              0
DEP_TIME                  147894
DEP_DELAY                 147918
CRS_ARR_TIME              0
ARR_TIME                  156916
ARR_DELAY                 174420
CANCELLED                 0
CANCELLATION_CODE         7938055
DIVERTED                  0
CRS_ELAPSED_TIME          10
ACTUAL_ELAPSED_TIME       174420
AIR_TIME                  174420
FLIGHTS                   0
DISTANCE                  0
CARRIER_DELAY             6564229
WEATHER_DELAY             6564229
NAS_DELAY                 6564229
SECURITY_DELAY            6564229
LATE_AIRCRAFT_DELAY       6564229
Unnamed: 29               8091684
dtype: int64
```

In [9]:
```
#Percentage of records missing in each variables

df_miss = df.isnull().sum()* 100 / len(df)
round(df_miss,2).sort_values(ascending = False)
```

Out[9]:
```
Unnamed: 29             100.00
CANCELLATION_CODE        98.10
SECURITY_DELAY           81.12
NAS_DELAY                81.12
WEATHER_DELAY            81.12
CARRIER_DELAY            81.12
LATE_AIRCRAFT_DELAY      81.12
ARR_DELAY                 2.16
AIR_TIME                  2.16
ACTUAL_ELAPSED_TIME       2.16
ARR_TIME                  1.94
DEP_DELAY                 1.83
DEP_TIME                  1.83
TAIL_NUM                  0.35
DIVERTED                  0.00
OP_CARRIER_FL_NUM         0.00
MONTH                     0.00
DAY_OF_MONTH              0.00
DAY_OF_WEEK               0.00
FL_DATE                   0.00
OP_UNIQUE_CARRIER         0.00
DEST                      0.00
ORIGIN                    0.00
CRS_ELAPSED_TIME          0.00
CRS_DEP_TIME              0.00
CRS_ARR_TIME              0.00
DISTANCE                  0.00
FLIGHTS                   0.00
CANCELLED                 0.00
YEAR                      0.00
dtype: float64
```

In [10]:
```
#Checking the basic statistical information

df.describe()
```

Out[10]:

	YEAR	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	OP_CARRIER_FL_NUM	CRS_DEP
count	8091684.0	8.091684e+06	8.091684e+06	8.091684e+06	8.091684e+06	8.09168
mean	2019.0	6.573170e+00	1.573118e+01	3.937864e+00	2.712855e+03	1.33028
std	0.0	3.402571e+00	8.762414e+00	1.995895e+00	1.836274e+03	4.90583
min	2019.0	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.00000
25%	2019.0	4.000000e+00	8.000000e+00	2.000000e+00	1.122000e+03	9.15000
50%	2019.0	7.000000e+00	1.600000e+01	4.000000e+00	2.316000e+03	1.32200
75%	2019.0	1.000000e+01	2.300000e+01	6.000000e+00	4.213000e+03	1.73500
max	2019.0	1.200000e+01	3.100000e+01	7.000000e+00	9.401000e+03	2.35900

In [11]:
```
#Removing Columns with many missing vlaues

Dr_Droped=df.drop(["CARRIER_DELAY","WEATHER_DELAY","NAS_DELAY","SECURITY_DE
```

In [12]:
```
#Copying the dataframe back to df

df = Dr_Droped
df
```

Out[12]:

	YEAR	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	OP_UNIQUE_CARRIER	TAIL_NUM	OP_(
0	2019	1	16	3	AA	N150UW	
1	2019	1	17	4	AA	N563UW	
2	2019	1	18	5	AA	N921US	
3	2019	1	19	6	AA	N604AW	
4	2019	1	20	7	AA	N975UY	
...	...	...	...	...	...	...	...
8091679	2019	12	26	4	AA	N947NN	
8091680	2019	12	27	5	AA	N962AN	
8091681	2019	12	28	6	AA	N946AN	
8091682	2019	12	29	7	AA	N801NN	
8091683	2019	12	30	1	AA	N843NN	

8091684 rows × 22 columns

In [13]:
```
#Percentage of records missing in each variables after removing removing th

df_miss = df.isnull().sum()* 100 / len(df)
round(df_miss,2).sort_values(ascending = False)
```

Out[13]:
```
AIR_TIME               2.16
ACTUAL_ELAPSED_TIME    2.16
ARR_DELAY              2.16
ARR_TIME               1.94
DEP_TIME               1.83
DEP_DELAY              1.83
TAIL_NUM               0.35
ORIGIN                 0.00
MONTH                  0.00
DAY_OF_MONTH           0.00
DAY_OF_WEEK            0.00
OP_UNIQUE_CARRIER      0.00
OP_CARRIER_FL_NUM      0.00
DISTANCE               0.00
DEST                   0.00
CRS_DEP_TIME           0.00
FLIGHTS                0.00
CRS_ARR_TIME           0.00
CANCELLED              0.00
DIVERTED               0.00
CRS_ELAPSED_TIME       0.00
YEAR                   0.00
dtype: float64
```

In [14]:
```
#Replacing Null values with mean of the variable

df.DEP_TIME.fillna(df.DEP_TIME.mean(),inplace=True)
df.DEP_TIME = df.DEP_TIME.astype(int)
```

In [15]:
```
#Replacing Null values with mean of the variable

df.DEP_DELAY.fillna(df.DEP_DELAY.mean(),inplace=True)
df.DEP_DELAY = df.DEP_DELAY.astype(int)
```

In [16]:
```
#Replacing Null values with mean of the variable

df.ARR_TIME.fillna(df.ARR_TIME.mean(),inplace=True)
df.ARR_TIME = df.ARR_TIME.astype(int)
```

In [17]:
```
#Replacing Null values with mean of the variable

df.ARR_DELAY.fillna(df.ARR_DELAY.mean(),inplace=True)
df.ARR_DELAY = df.ARR_DELAY.astype(int)
```

In [18]:
```
#Replacing Null values with mean of the variable

df.ACTUAL_ELAPSED_TIME.fillna(df.ACTUAL_ELAPSED_TIME.mean(),inplace=True)
df.ACTUAL_ELAPSED_TIME = df.ACTUAL_ELAPSED_TIME.astype(int)
```

In [19]:
```
#Replacing Null values with mean of the variable

df.AIR_TIME.fillna(df.AIR_TIME.mean(),inplace=True)
df.AIR_TIME = df.AIR_TIME.astype(int)
```

In [20]:
```
#Replacing Null values with mean of the variable

df.CRS_ELAPSED_TIME.fillna(df.CRS_ELAPSED_TIME.mean(),inplace=True)
df.CRS_ELAPSED_TIME = df.CRS_ELAPSED_TIME.astype(int)
```

In [21]:
```
# Replacing the null values for categorical data with "unknown"

df['TAIL_NUM'] = df['TAIL_NUM'].fillna('NA')
```

In [22]:
```
#Checking Null values after data imputation

df.isnull().sum()
```

Out[22]:
```
YEAR                 0
MONTH                0
DAY_OF_MONTH         0
DAY_OF_WEEK          0
OP_UNIQUE_CARRIER    0
TAIL_NUM             0
OP_CARRIER_FL_NUM    0
ORIGIN               0
DEST                 0
CRS_DEP_TIME         0
DEP_TIME             0
DEP_DELAY            0
CRS_ARR_TIME         0
ARR_TIME             0
ARR_DELAY            0
CANCELLED            0
DIVERTED             0
CRS_ELAPSED_TIME     0
ACTUAL_ELAPSED_TIME  0
AIR_TIME             0
FLIGHTS              0
DISTANCE             0
dtype: int64
```

In [23]:
```
#Converting all the categorical values to Numerical Values

from sklearn import preprocessing

# limit to categorical data using df.select_dtypes()
Cat = df.select_dtypes(include=[object])
Cat.head(3)

# TODO: create a LabelEncoder object and fit it to each feature in X

# 1. INSTANTIATE
# encode labels with value between 0 and n_classes-1.
le = preprocessing.LabelEncoder()

# 2/3. FIT AND TRANSFORM
# use df.apply() to apply le.fit_transform to all columns
CatToInt = Cat.apply(le.fit_transform)
CatToInt.head()

Endf = df.drop(Cat.columns, inplace = False, axis='columns')

Endf = pd.concat([Endf, CatToInt], axis=1)
Endf.head(2)
```

Out[23]:

	YEAR	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	OP_CARRIER_FL_NUM	CRS_DEP_TIME	DEP_T
0	2019	1	16	3	544	537	
1	2019	1	17	4	544	537	

In [29]:
```
#Checking null values after converting into numerical data

Endf.isnull().sum()
```

Out[29]:
```
YEAR                   0
MONTH                  0
DAY_OF_MONTH           0
DAY_OF_WEEK            0
OP_CARRIER_FL_NUM      0
CRS_DEP_TIME           0
DEP_TIME               0
DEP_DELAY              0
CRS_ARR_TIME           0
ARR_TIME               0
ARR_DELAY              0
CANCELLED              0
DIVERTED               0
CRS_ELAPSED_TIME       0
ACTUAL_ELAPSED_TIME    0
AIR_TIME               0
FLIGHTS                0
DISTANCE               0
OP_UNIQUE_CARRIER      0
TAIL_NUM               0
ORIGIN                 0
DEST                   0
dtype: int64
```

In [31]:
```
#Dependent Variables

X = Endf.drop(["CANCELLED"],axis=1)
X
```

Out[31]:

	YEAR	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	OP_CARRIER_FL_NUM	CRS_DEP_TIME
0	2019	1	16	3	544	537
1	2019	1	17	4	544	537
2	2019	1	18	5	544	537
3	2019	1	19	6	544	537
4	2019	1	20	7	544	537
...	...	...	...	...	...	...
8091679	2019	12	26	4	1583	1820
8091680	2019	12	27	5	1583	1820
8091681	2019	12	28	6	1583	1820
8091682	2019	12	29	7	1583	1820
8091683	2019	12	30	1	1583	1820

8091684 rows × 21 columns

In [32]:
```
#Independent Variable

y = Endf["CANCELLED"]
y
```

Out[32]:
```
0          0.0
1          0.0
2          0.0
3          0.0
4          1.0
           ...
8091679    0.0
8091680    0.0
8091681    0.0
8091682    0.0
8091683    0.0
Name: CANCELLED, Length: 8091684, dtype: float64
```

In [33]:
```
# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.7,
```

In [ ]:
```
#pd.crosstab(y_test, y_pred, rownames=['True'], colnames=['Predicted'], mar
```

Decision Tree

In [34]:
```
from sklearn import tree
df_Class = tree.DecisionTreeClassifier()
df_Class = df_Class.fit(X_train, y_train)
```

In [35]:
```
y_pred=df_Class.predict(X_test)
```

In [36]:
```
from sklearn.metrics import accuracy_score
accuracy=accuracy_score(y_pred, y_test)
print('LightGBM Model accuracy score: {0:0.4f}'.format(accuracy_score(y_tes
```

```
LightGBM Model accuracy score: 1.0000
```

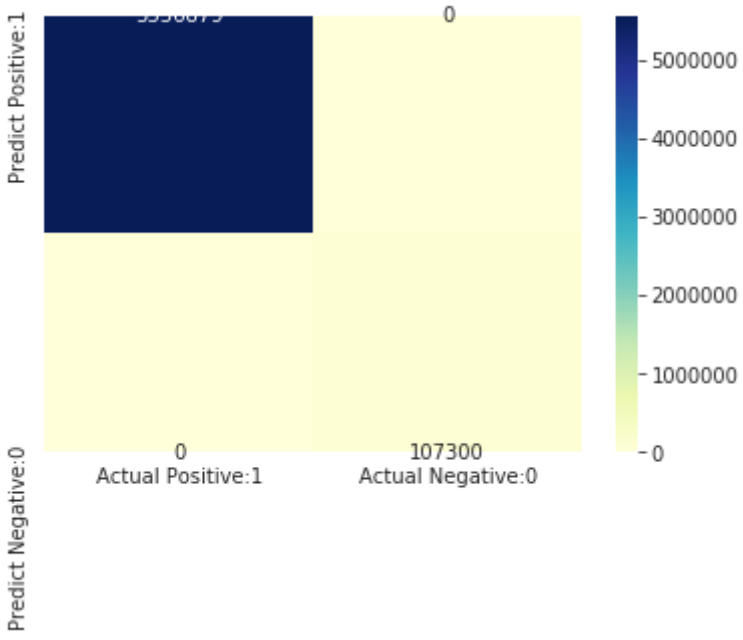
In [37]:
```
print('Decision Tree Classifier')
print(confusion_matrix(y_test, df_Class.predict(X_test)))
print(classification_report(y_test, y_pred))
```

```
Decision Tree Classifier
[[5556879       0]
 [      0  107300]]
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00   5556879
         1.0       1.00      1.00      1.00    107300

    accuracy                           1.00   5664179
   macro avg       1.00      1.00      1.00   5664179
weighted avg       1.00      1.00      1.00   5664179
```

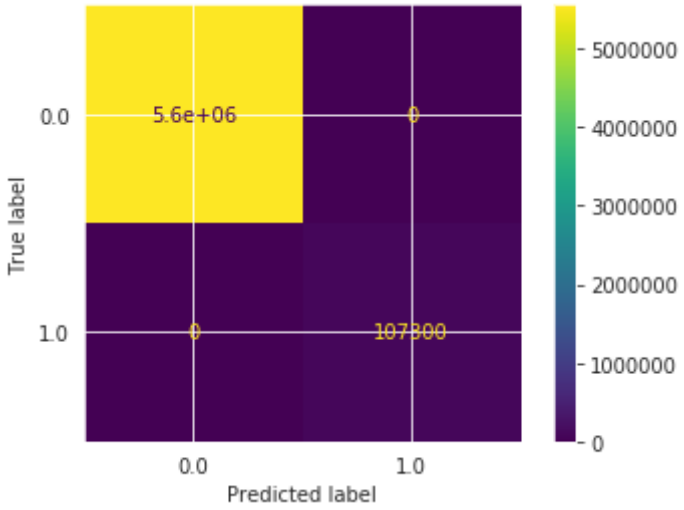
In [65]:
```
print('Decision Tree Classifier')
cm_matrix = pd.DataFrame(data=confusion_matrix(y_test, df_Class.predict(X_t
                                   index=['Predict Positive:1', 'Predict Nega
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu', )
```

```
Decision Tree Classifier
```

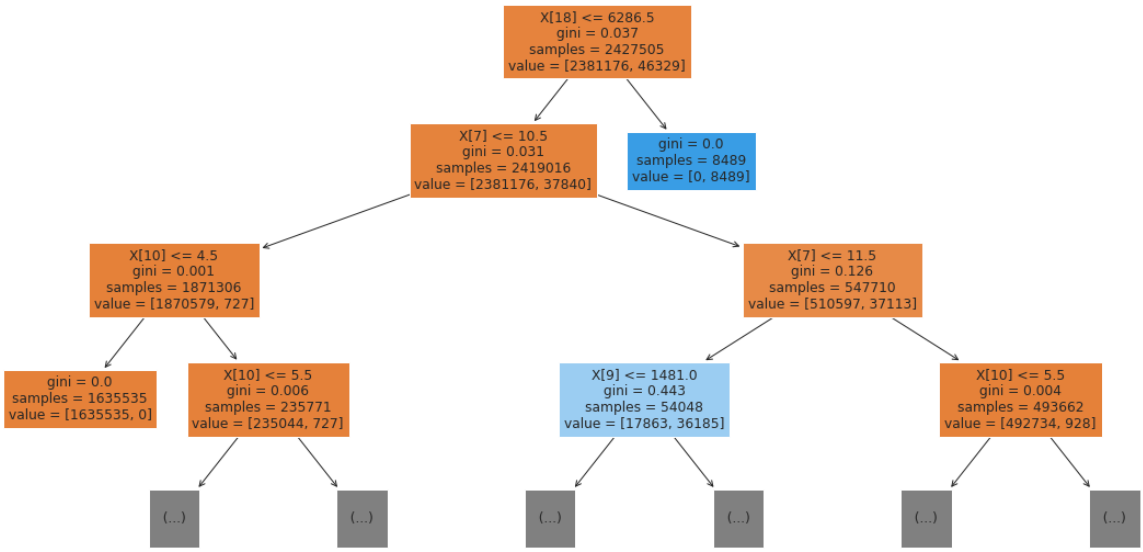
Out[65]: <matplotlib.axes._subplots.AxesSubplot at 0x2ace01d4a7d0>



In [70]:
```
plot_confusion_matrix(df_Class, X_test, y_test)
plt.show()
```



In [45]:
```
fig = plt.figure(figsize=(20,10))

_ = tree.plot_tree(df_Class, filled=True,max_depth = 3)
```



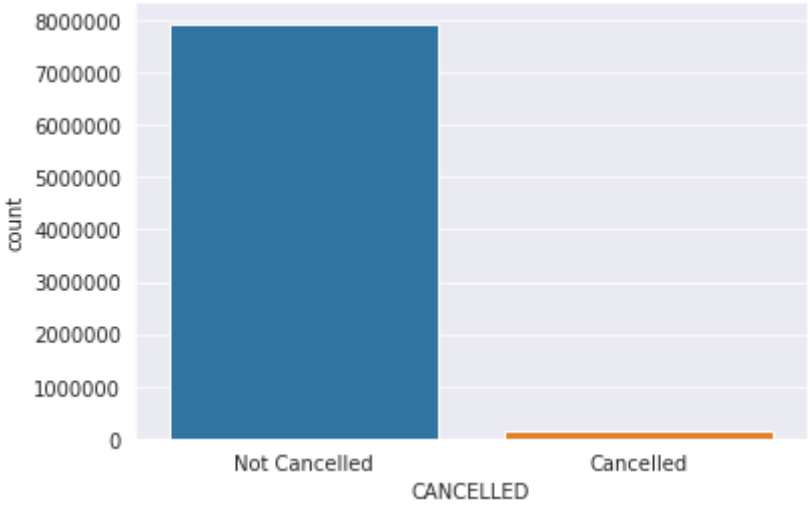
In [46]:
```
from sklearn.feature_selection import SequentialFeatureSelector
```

In [120]:
```
sfs = SequentialFeatureSelector(df_Class, n_features_to_select=3)
sfs.fit(X_train, y_train)
sfs.get_feature_names_out()
```

Out[120]:
```
array(['YEAR', 'ARR_TIME', 'DIVERTED'], dtype=object)
```

New

In [48]:
```
#Visualising Data Imbalance

g = sns.countplot(df['CANCELLED'])
g.set_xticklabels(['Not Cancelled', 'Cancelled'])
plt.show()
```



Random Forest

In [49]:
```
#Loading the required library

from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier,RandomForestClassif
```

In [ ]:
```
# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.7,
```

In [ ]:
```
#Checking the shape of the train and test

print(f"Train data shape:{X_train.shape}")
print(f"Test data shape:{X_test.shape}")
```

In [ ]:
```
X_train.head()
```

In [40]:
```
Endf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8091684 entries, 0 to 8091683
Data columns (total 22 columns):
YEAR                   int64
MONTH                  int64
DAY_OF_MONTH           int64
DAY_OF_WEEK            int64
OP_CARRIER_FL_NUM      int64
CRS_DEP_TIME           int64
DEP_TIME               int64
DEP_DELAY              int64
CRS_ARR_TIME           int64
ARR_TIME               int64
ARR_DELAY              int64
CANCELLED              float64
DIVERTED               float64
CRS_ELAPSED_TIME       int64
ACTUAL_ELAPSED_TIME    int64
AIR_TIME               int64
FLIGHTS                float64
DISTANCE               float64
OP_UNIQUE_CARRIER      int64
TAIL_NUM               int64
ORIGIN                 int64
DEST                   int64
dtypes: float64(4), int64(18)
memory usage: 1.3 GB
```

In [41]:
```
#Checking Null values after data imputation

Endf.isnull().sum()
```

Out[41]:
```
YEAR                   0
MONTH                  0
DAY_OF_MONTH           0
DAY_OF_WEEK            0
OP_CARRIER_FL_NUM      0
CRS_DEP_TIME           0
DEP_TIME               0
DEP_DELAY              0
CRS_ARR_TIME           0
ARR_TIME               0
ARR_DELAY              0
CANCELLED              0
DIVERTED               0
CRS_ELAPSED_TIME       0
ACTUAL_ELAPSED_TIME    0
AIR_TIME               0
FLIGHTS                0
DISTANCE               0
OP_UNIQUE_CARRIER      0
TAIL_NUM               0
ORIGIN                 0
DEST                   0
dtype: int64
```

In [36]:
```
#Endf.AIR_TIME.fillna(method="ffill", inplace= True)
```

In [82]:
```
X = Endf.drop(["CANCELLED"],axis=1)
X
```

Out[82]:

	YEAR	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	OP_CARRIER_FL_NUM	CRS_DEP_TIME
4	2019	1	20	7	544	537
5	2019	1	21	1	544	537
6	2019	1	22	2	544	537
27	2019	1	12	6	545	2046
28	2019	1	13	7	545	2046
...	...	...	...	...	...	...
7134475	2019	11	16	6	4954	1529
6852771	2019	11	3	7	882	1508
2612300	2019	5	29	3	4925	1708
1414926	2019	3	15	5	2967	840
4558506	2019	7	14	7	5406	2031

307258 rows × 21 columns

In [83]:
```
y = Endf["CANCELLED"]
y
```

Out[83]:
```
4          1.0
5          1.0
6          1.0
27         1.0
28         1.0
          ...
7134475    0.0
6852771    0.0
2612300    0.0
1414926    0.0
4558506    0.0
Name: CANCELLED, Length: 307258, dtype: float64
```

In [84]:
```
# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.7,
```

In [50]:
```
# run random forest to get feature importance

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators = 5).fit(X_train, y_train)

feats = X_train.columns

for feature in zip(feats, rf.feature_importances_):
    print(feature)
```

```
('YEAR', 0.0)
('MONTH', 0.0002264502692303077)
('DAY_OF_MONTH', 0.00019021442561764057)
('DAY_OF_WEEK', 0.00010898460194011015)
('OP_CARRIER_FL_NUM', 0.0005644720572442424)
('CRS_DEP_TIME', 0.002374702621107596)
('DEP_TIME', 0.2240378968035776)
('DEP_DELAY', 0.1734768545504583)
('CRS_ARR_TIME', 0.0003047609753749788)
('ARR_TIME', 0.09899693643514466)
('ARR_DELAY', 0.13166316188515792)
('DIVERTED', 0.009864816562058306)
('CRS_ELAPSED_TIME', 0.001532190516132091)
('ACTUAL_ELAPSED_TIME', 0.28980624301928093)
('AIR_TIME', 0.00857353831443399)
('FLIGHTS', 0.0)
('DISTANCE', 0.0015505526253696016)
('OP_UNIQUE_CARRIER', 0.00043654373376373696)
('TAIL_NUM', 0.05561314806262306)
('ORIGIN', 0.0003258484831479049776)
('DEST', 0.0003526842266941040 5)
```

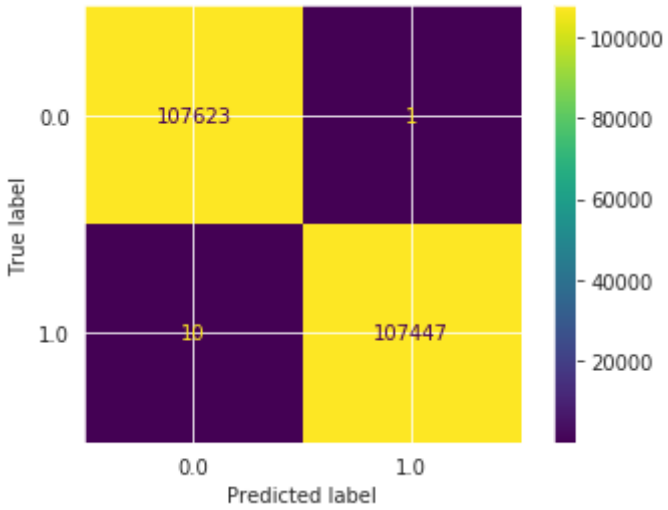
In [ ]:
```
# pip install imbalanced-learn --force-reinstall --user
```

In [51]:
```
y_pred_rf = rf.predict(X_test)
```

In [52]:
```
print ('Accuracy: ', accuracy_score(y_test, y_pred_rf))
print ('F1 score: ', f1_score(y_test, y_pred_rf))
print ('Recall: ', recall_score(y_test, y_pred_rf))
print ('Precision: ', precision_score(y_test, y_pred_rf))
print ('\n clasification report:\n', classification_report(y_test,y_pred_rf
print ('\n confussion matrix:\n',confusion_matrix(y_test, y_pred_rf))
```

```
Accuracy:  0.9999934677205646
F1 score:  0.9998275950440565
Recall:  0.999878844361603
Precision:  0.9997763509798623

 clasification report:
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00   5556879
         1.0       1.00      1.00      1.00    107300

    accuracy                           1.00   5664179
   macro avg       1.00      1.00      1.00   5664179
weighted avg       1.00      1.00      1.00   5664179


 confussion matrix:
 [[5556855      24]
 [     13  107287]]
```

In [106]:
```
plot_confusion_matrix(rf, X_test, y_test)
plt.show()
```



In [54]:
```
from sklearn.feature_selection import SequentialFeatureSelector
```

In [119]:
```
sfs = SequentialFeatureSelector(rf, n_features_to_select=3)
sfs.fit(X_train, y_train)
sfs.get_feature_names_out()
```

Out[119]:
```
array(['YEAR', 'ARR_TIME', 'DIVERTED'], dtype=object)
```

Random Undersampling

In [71]:
```
# Random Undersampling

from imblearn.under_sampling import RandomUnderSampler

under_sample = RandomUnderSampler(random_state = 5)
X_resampled_us, y_resampled_us = under_sample.fit_resample(X_train, y_train
len(X_resampled_us)
```

Out[71]:
```
92658
```

In [72]:
```
from collections import Counter

print(sorted(Counter(y_resampled_us).items()))
```

```
[(0.0, 46329), (1.0, 46329)]
```

In [73]:
```
# Random Forest - Random UnderSampling

rf_us = RandomForestClassifier()
rf_us.fit(X_resampled_us, y_resampled_us)
```

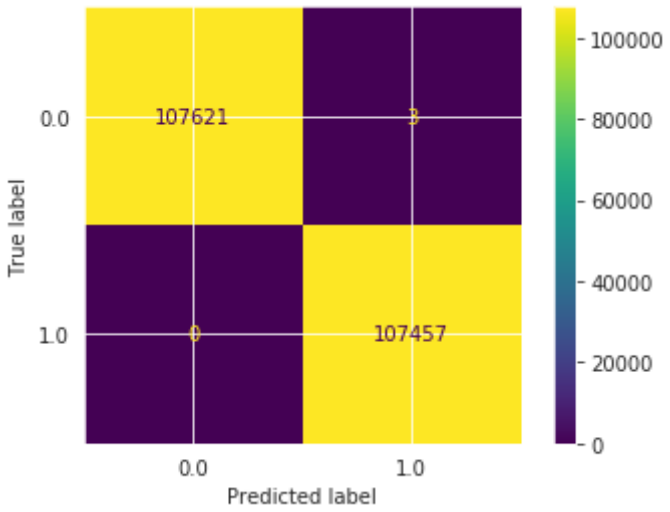
Out[73]:
```
RandomForestClassifier()
```

In [59]:
```
y_pred_rf_us = rf_us.predict(X_test)
```

In [60]:
```
print ('Accuracy: ', accuracy_score(y_test, y_pred_rf_us))
print ('F1 score: ', f1_score(y_test, y_pred_rf_us))
print ('Recall: ', recall_score(y_test, y_pred_rf_us))
print ('Precision: ', precision_score(y_test, y_pred_rf_us))
print ('\n clasification report:\n', classification_report(y_test,y_pred_rf
print ('\n confussion matrix:\n',confusion_matrix(y_test, y_pred_rf_us))
```

```
Accuracy:  0.9999931146243789
F1 score:  0.999818299563453
Recall:  1.0
Precision:  0.9996366651450078

 clasification report:
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00   5556879
         1.0       1.00      1.00      1.00    107300

    accuracy                           1.00   5664179
   macro avg       1.00      1.00      1.00   5664179
weighted avg       1.00      1.00      1.00   5664179


 confussion matrix:
 [[5556840      39]
 [      0  107300]]
```

In [107]:
```
plot_confusion_matrix(rf_us, X_test, y_test)
plt.show()
```



In [118]:
```
sfs = SequentialFeatureSelector(rf_us, n_features_to_select=3)
sfs.fit(X_train, y_train)
sfs.get_feature_names_out()
```

Out[118]:
```
array(['YEAR', 'ARR_TIME', 'DIVERTED'], dtype=object)
```

Random Oversampling

In [110]:
```
# Random OverSampling

from imblearn.over_sampling import RandomOverSampler

over_sample = RandomOverSampler(sampling_strategy = 1)
X_resampled_os, y_resampled_os = over_sample.fit_resample(X_train, y_train)
len(X_resampled_os)
```

Out[110]:
```
92344
```

In [116]:
```
from collections import Counter

print(sorted(Counter(y_resampled_os).items()))
```

```
[(0.0, 46172), (1.0, 46172)]
```

In [111]:
```
# Random Forest - Random Over-Sampling

rf_os = RandomForestClassifier()
rf_os.fit(X_resampled_os, y_resampled_os)
```

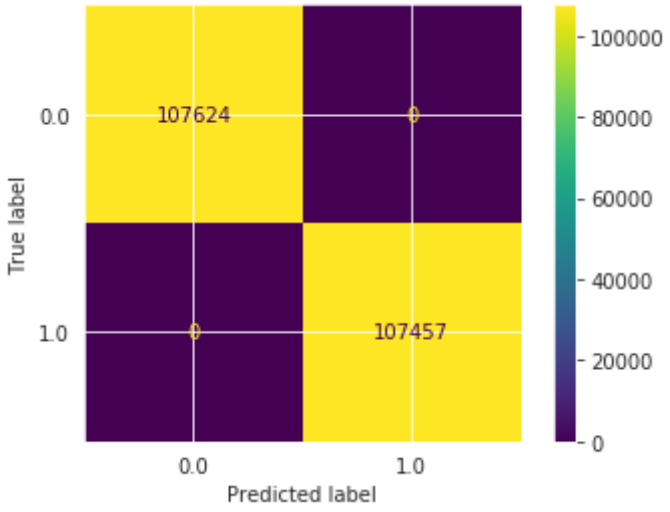
Out[111]:
```
RandomForestClassifier()
```

In [103]:
```
y_pred_rf_os = rf_os.predict(X_test)
```

In [112]:
```
print ('Accuracy: ', accuracy_score(y_test, y_pred_rf_os))
print ('F1 score: ', f1_score(y_test, y_pred_rf_os))
print ('Recall: ', recall_score(y_test, y_pred_rf_os))
print ('Precision: ', precision_score(y_test, y_pred_rf_os))
print ('\n clasification report:\n', classification_report(y_test,y_pred_rf
print ('\n confussion matrix:\n',confusion_matrix(y_test, y_pred_rf_os))
```

```
Accuracy:  1.0
F1 score:  1.0
Recall:  1.0
Precision:  1.0

 clasification report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00    107624
         1.0       1.00      1.00      1.00    107457

    accuracy                           1.00    215081
   macro avg       1.00      1.00      1.00    215081
weighted avg       1.00      1.00      1.00    215081


 confussion matrix:
 [[107624      0]
 [     0 107457]]
```

In [113]:
```
plot_confusion_matrix(rf_os, X_test, y_test)
plt.show()
```



In [117]:
```
sfs = SequentialFeatureSelector(rf_os, n_features_to_select=3)
sfs.fit(X_train, y_train)
sfs.get_feature_names_out()
```

Out[117]:
```
array(['YEAR', 'ARR_TIME', 'DIVERTED'], dtype=object)
```

By Taking Two differnt Datasets and Merging

In [90]:
```
Sample_Data = Endf[Endf.CANCELLED==0].sample(n = 153629)
Sample_Data
```

Out[90]:

	YEAR	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	OP_CARRIER_FL_NUM	CRS_DEP_TIME
1270392	2019	3	31	7	1410	1940
1290204	2019	3	27	3	673	1818
6419259	2019	10	21	1	3012	1344
8070957	2019	12	3	2	16	2210
7963116	2019	12	17	2	3197	2215
...	...	...	...	...	...	...
3978501	2019	7	21	7	57	2021
6813557	2019	11	8	5	1825	2309
6585225	2019	10	18	5	323	1903
2014226	2019	4	30	2	230	2258
3121658	2019	5	10	5	1869	1500

153629 rows × 22 columns

In [91]:
```
Sample_Data_1 = Endf[Endf.CANCELLED==1]
```

In [92]:
```
Endf_1 = pd.concat([Sample_Data_1,Sample_Data])
Endf_1
```

Out[92]:

	YEAR	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	OP_CARRIER_FL_NUM	CRS_DEP_TIME
4	2019	1	20	7	544	537
5	2019	1	21	1	544	537
6	2019	1	22	2	544	537
27	2019	1	12	6	545	2046
28	2019	1	13	7	545	2046
...	...	...	...	...	...	...
3978501	2019	7	21	7	57	2021
6813557	2019	11	8	5	1825	2309
6585225	2019	10	18	5	323	1903
2014226	2019	4	30	2	230	2258
3121658	2019	5	10	5	1869	1500

307258 rows × 22 columns

In [93]:
```
Endf_1.CANCELLED.value_counts()
```

Out[93]:
```
0.0    153629
1.0    153629
Name: CANCELLED, dtype: int64
```

In [94]:
```
#Dependent Variable

X = Endf_1.drop(["CANCELLED"],axis=1)
X
```

Out[94]:

	YEAR	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	OP_CARRIER_FL_NUM	CRS_DEP_TIME
4	2019	1	20	7	544	537
5	2019	1	21	1	544	537
6	2019	1	22	2	544	537
27	2019	1	12	6	545	2046
28	2019	1	13	7	545	2046
...	...	...	...	...	...	...
3978501	2019	7	21	7	57	2021
6813557	2019	11	8	5	1825	2309
6585225	2019	10	18	5	323	1903
2014226	2019	4	30	2	230	2258
3121658	2019	5	10	5	1869	1500

307258 rows × 21 columns

In [95]:
```
#Independent Variable

y = Endf_1["CANCELLED"]
y
```

Out[95]:
```
4          1.0
5          1.0
6          1.0
27         1.0
28         1.0
          ...
3978501    0.0
6813557    0.0
6585225    0.0
2014226    0.0
3121658    0.0
Name: CANCELLED, Length: 307258, dtype: float64
```

Random Forest after Merging two Samples of Data

In [96]:
```
# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.7,
```

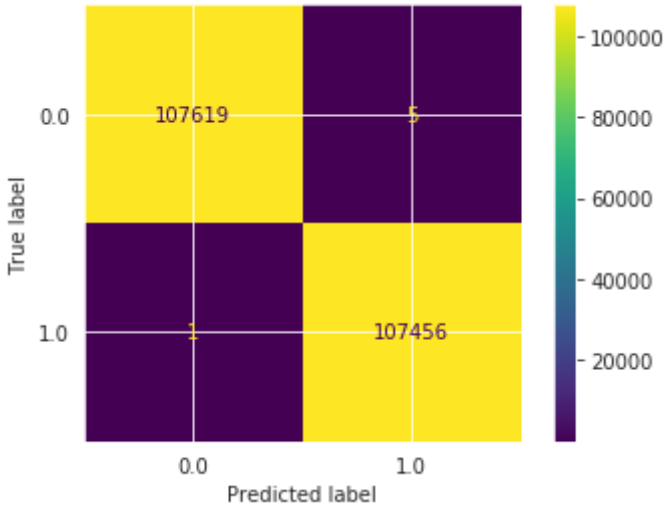
In [97]:
```
#Random Forest

from sklearn.ensemble import RandomForestClassifier

rf_1 = RandomForestClassifier(n_estimators = 5).fit(X_train, y_train)

feats_1 = X_train.columns
```

In [98]:
```
y_pred_rf_1 = rf.predict(X_test)
```

In [99]:
```
print ('Accuracy: ', accuracy_score(y_test, y_pred_rf_1))
print ('F1 score: ', f1_score(y_test, y_pred_rf_1))
print ('Recall: ', recall_score(y_test, y_pred_rf_1))
print ('Precision: ', precision_score(y_test, y_pred_rf_1))
print ('\n clasification report:\n', classification_report(y_test,y_pred_rf
print ('\n confussion matrix:\n',confusion_matrix(y_test, y_pred_rf_1))
```

```
Accuracy:  0.9999488564773271
F1 score:  0.9999488145924944
Recall:  0.9999069395199941
Precision:  0.9999906931725113

 clasification report:
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00    107624
         1.0       1.00      1.00      1.00    107457

    accuracy                           1.00    215081
   macro avg       1.00      1.00      1.00    215081
weighted avg       1.00      1.00      1.00    215081


 confussion matrix:
 [[107623      1]
 [    10 107447]]
```

In [114]:
```
plot_confusion_matrix(rf_1, X_test, y_test)
plt.show()
```



In [100]:
```
sfs = SequentialFeatureSelector(rf_1, n_features_to_select=3)
sfs.fit(X_train, y_train)
sfs.get_feature_names_out()
```

Out[100]:
```
array(['YEAR', 'ARR_TIME', 'DIVERTED'], dtype=object)
```

In [ ]:
```

```