Statistical Learning and Data Modelling

Mini Project

Modeling Flight Delay

Amirta V - 21060641004
Durgesh Kinnerkar - 21060641016
Joy Angelin - 21060641022
Ravi Shankar - 21060641037

Table of Contents

Executive Summary

The project's main objective is to study and understand the factors influencing flight cancellation and flight delay based on historical flight data. The primary objectives can be summarized as follows:

1. Predicting which flights will be canceled or delayed
2. Predicting the delay time

The approach towards analysis and modeling to predict flight cancellation and flight delay is based on understanding the data and the significance of the attributes. An intuitive understanding of the attributes and their interpretation of the business problem is used to develop the predictive models for classification and regression. After conventional data preprocessing operations such as missing value imputation, data cleaning, and label encoding are carried out, a couple of new attributes are created to efficiently classify instances into two groups - flights that are delayed or canceled and flights that are on time. A decision tree classifier was developed for feature selection. Three classifiers are created for the classification task: Random Forest Classifier, Logistic Regression, and Gaussian Naive Bayes. The performance of the models was evaluated using metrics such as confusion matrix, classification report, and ROC Curve (for logistic regression). The Random Forest Classifier has the highest accuracy, with 77%.

For predicting the delay time, the approach was to show the importance of variable selection in modeling. To achieve this, intuitive and decision tree-based methods were used to select the variable. Multiple Linear Regression, Random Forest Regression, and XGBoost regression models were developed for intuitively selected variables. The XGBoost regression model was the best, with 41.54 lowest RMSE value among the three. Random Forest Regressor and XGBoost regressor models were developed for the variables selected using the decision tree technique. Among these, Random forest regression has the least 39.55 RMSE value.

Business Problem :

Objective: The study's main aim is to develop a model to understand the pattern of the flight timings in the data and predict whether a flight will be delayed or canceled and subsequently predict the delay time based on historical flight data.

Flight schedule planning is a significant challenge in the airline industry as it is exposed to many uncertain conditions. One of the significant challenges is delay occurrence and flight cancellations. Departure delays can occur due to technical issues, weather conditions, air traffic, airport facilities, and other factors. An airline company is mainly interested in avoiding the issue

20s created by scheduling flights and operational management due to flight delays and cancellations, which can create client discontent.

A delay prediction tool that ranks the attributes according to their significance in affecting flight delay and cancellation will allow airlines and airport management to take actions for smooth operations, inform concerned stakeholders (ground staff, airport management, passengers), and avoid passenger discontent. This prediction of flight delay/cancellation enables airlines to respond to the delays' potential causes in advance and lessen the impact. It also enables passengers to be well prepared for the inconvenience caused to their journey.

Data Description:

- Name of the dataset : Airline On-Time database from the TranStats data library.
- Number of data points : 1048575
- Number of attributes : 61
- Description:
    - The dataset provides the status of flights from 27 airlines for January and October 2018.
    - The dataset provides complete information on the flight from its take-off to its landing. The information includes the names of the airline, origin, destination, diversion of flight (number of times), the time associated with each taxi in, taxi out, wheels on, wheels off, arrival time, Departure time, flight in the air, delay in arrival or departure (if any).
    - For a better understanding, the data provides information on the flight number, airport code, etc.

Exploratory Data Analysis:

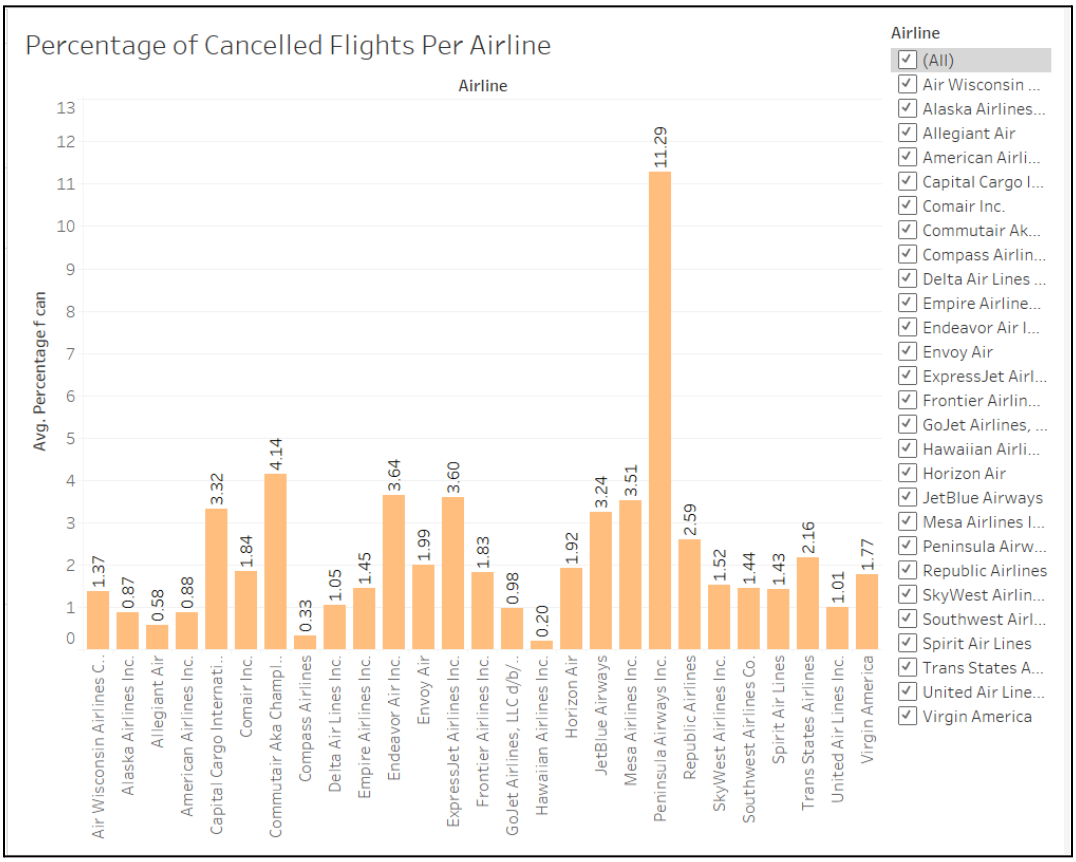
Visualizations

1. Summary of Delayed flights from different airlines

Airline	min	max	count	mean
Peninsula Airways Inc.	-80.0	477.0	311.0	16.106109
Cape Air	-95.0	260.0	340.0	-7.288235
Virgin America	-91.0	1024.0	5715.0	0.070866
Capital Cargo International	-72.0	891.0	6493.0	10.331126
Commutair Aka Champlain Enterprises, Inc.	-65.0	2482.0	6679.0	35.620153
Trans States Airlines	-62.0	3056.0	10735.0	32.383791
Hawaiian Airlines Inc.	-71.0	3399.0	12217.0	1.871245
Compass Airlines	-2570.0	4135.0	12474.0	14.071589
Air Wisconsin Airlines Corp	-70.0	2457.0	13668.0	10.998098
Allegiant Air	-89.0	1341.0	14289.0	14.341801
GoJet Airlines, LLC d/b/a United Express	-1086.0	2263.0	14639.0	21.251657
Horizon Air	-80.0	1277.0	17577.0	6.405302
Mesa Airlines Inc.	-81.0	3496.0	19501.0	22.559971
Frontier Airlines Inc.	-88.0	2507.0	19827.0	28.720129
Comair Inc.	-85.0	2213.0	23461.0	6.691104
Envoy Air	-78.0	1793.0	25650.0	7.871813
Spirit Air Lines	-88.0	2740.0	27524.0	2.648598
ExpressJet Airlines Inc.	-78.0	2865.0	28343.0	21.529584
Endeavor Air Inc.	-88.0	3837.0	29884.0	12.033530
Republic Airlines	-81.0	2616.0	36375.0	9.315821
Alaska Airlines Inc.	-96.0	1088.0	36520.0	-1.331188
JetBlue Airways	-84.0	2962.0	48046.0	23.380448
American Airlines Inc.	-67.0	4262.0	64058.0	16.233132
United Air Lines Inc.	-81.0	2847.0	93645.0	6.972054
SkyWest Airlines Inc.	-109.0	2862.0	101775.0	16.648430
Delta Air Lines Inc.	-90.0	2370.0	151550.0	3.653336
Southwest Airlines Co.	-122.0	1277.0	206381.0	9.475378

From the table, **Southwest Airlines Co**. had the highest number of flights while **Peninsula Airways Inc.** had the least. **Commutair Aka Champlain Enterprises, Inc.** had the highest average delay in minutes.

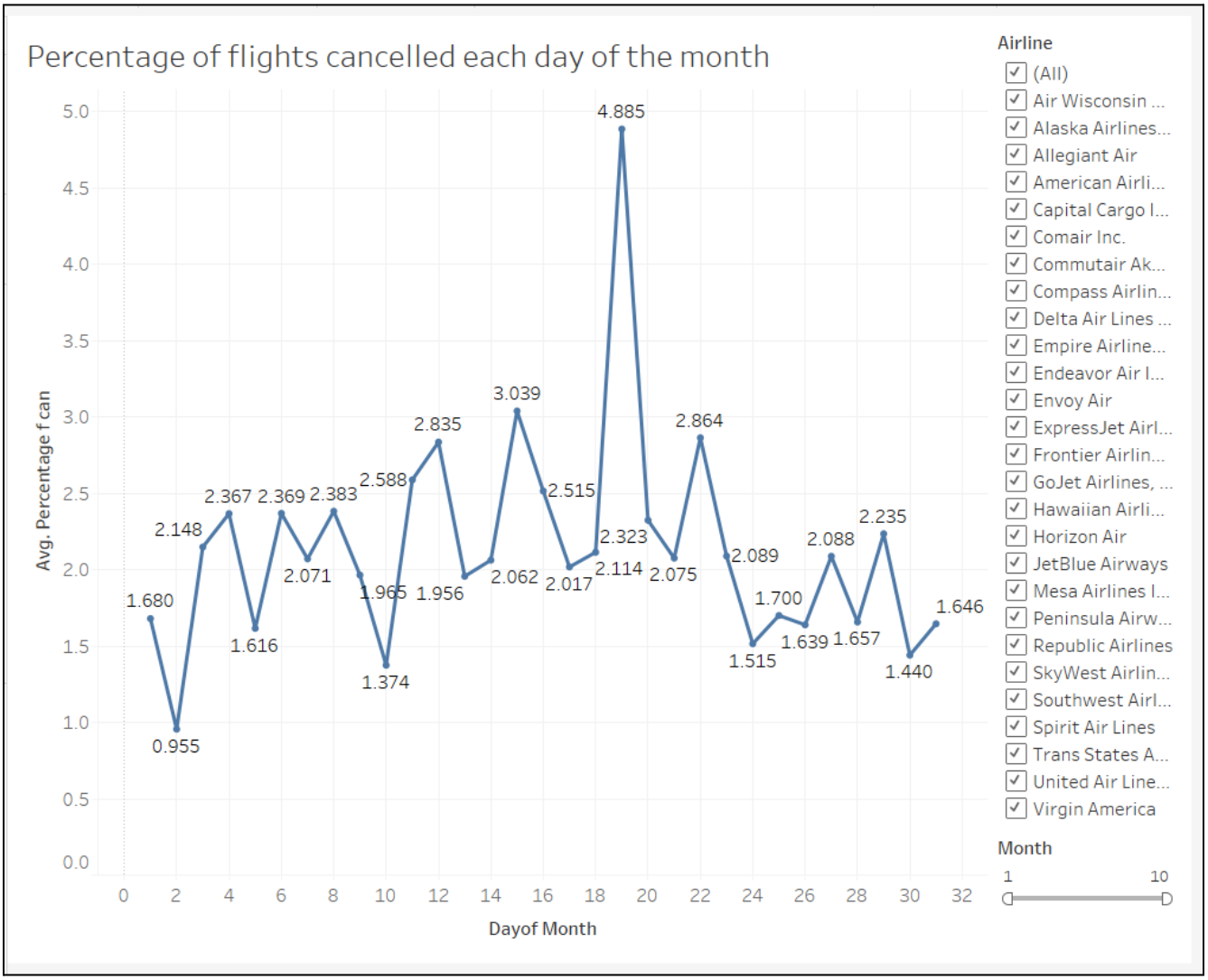
2. Percentage of Flights Canceled per Airline



The above Chart represents; out of the total flights from the particular Airline (flying from different origins), the given percentage of flights got canceled by that Airline.

The percentage of flights getting canceled is highest for **'Peninsula Airways Inc.'** (11.29%); therefore, flights of 'Peninsula Airways Inc.' have the highest probability of getting canceled. The percentage of flights getting canceled is lowest for **'Hawaiian Airlines Inc.'** (0.20%); therefore, flights of 'Hawaiian Airlines Inc.' have the lowest probability of getting canceled.

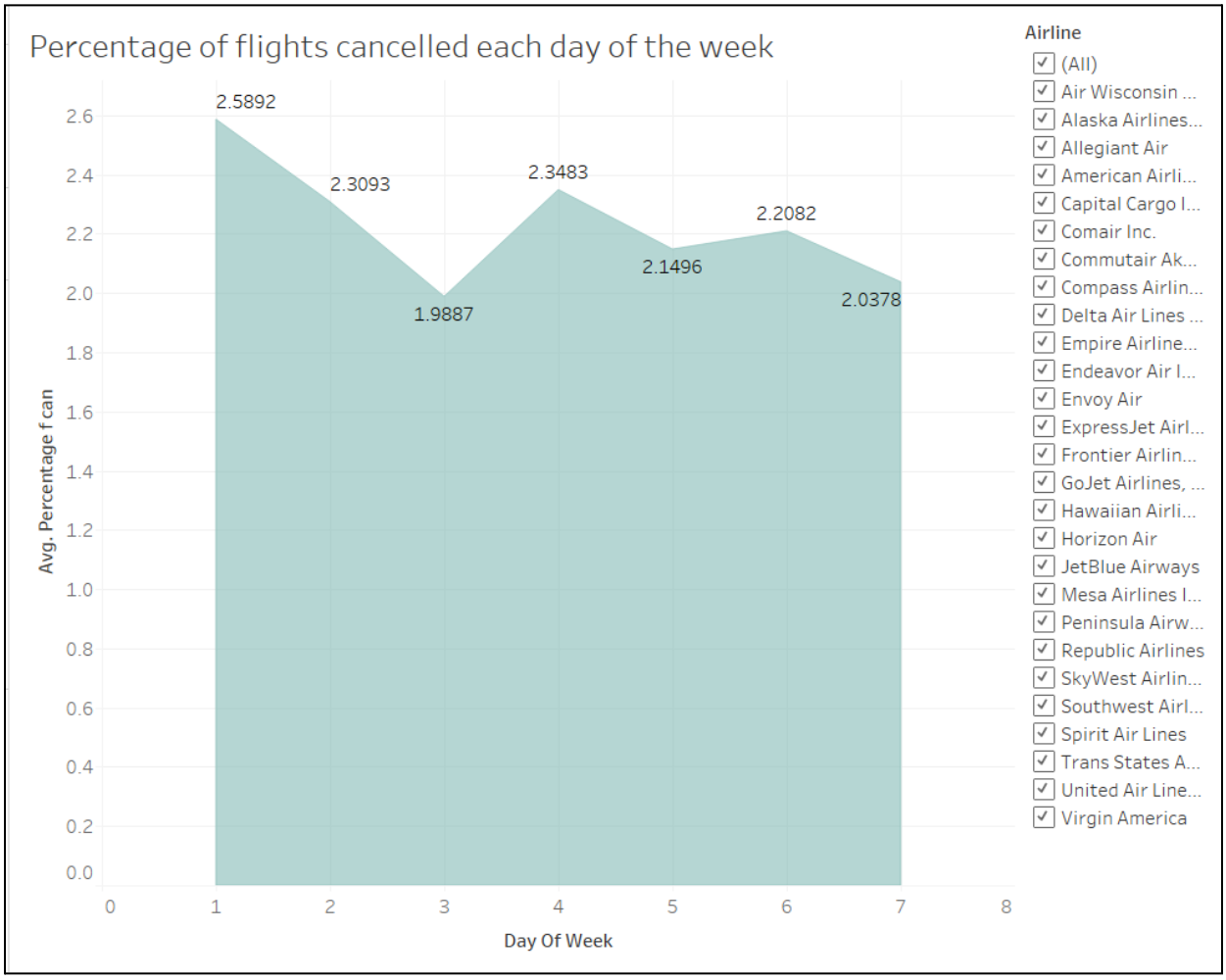
3. Percentage of Flights Canceled each day of the month



These are the average number of flights canceled in January 2018 and October 2018, considering all the Airlines. This can be calculated for individual airlines and individual months too.

Most flights were canceled on the **17th.** The least number of cancellations was on the **2nd.**

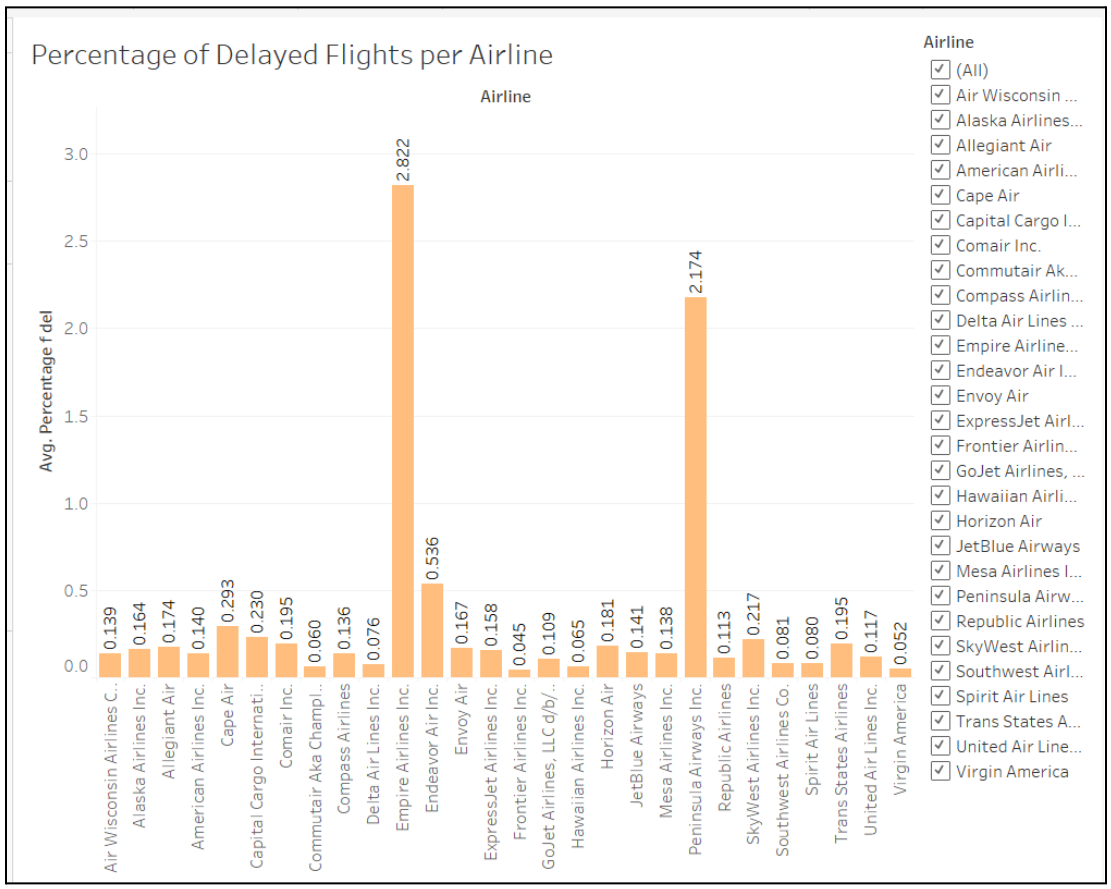
4. Percentage of Flights Canceled each day of the week



These are the average number of flights canceled for each day of the week in January 2018 and October 2018, considering all the Airlines. This can be calculated for individual airlines and individual months too.

Most flights were canceled on the **first day of the week (it could be a weekend).** The least number of cancellations was on the **third day of the week.**

5. Percentage of Flights Delayed per Airline



The above Chart represents; out of the total flights that belonged to a particular Airline (departing from different origins), the given percentage of flights got delayed from that Airline(either during departure or during arrival)

The percentage of flights getting delayed is highest for **'Empire Airlines Inc.'** (2.822%); therefore, flights of Empire Airlines Inc.' had the highest probability of getting delayed. The percentage of flights getting delayed is lowest for **'Frontier Airline INc.'** (0.045%); therefore, flights of Frontier Airline Inc.' had the lowest probability of getting delayed.

Pre Processing:

1) Classification Model

- Missing Values Treatment

Handling missing values is a critical part of data preprocessing. This can be attributed to the fact that it becomes difficult to perform various statistical tests and procedures. This also creates a hindrance in the implementation of various machine learning algorithms. Here, a few techniques have been introduced to tackle the problem of missing value.
The first is dropping variables with missing values. The second is intuitively selecting columns/variables that were not significant.

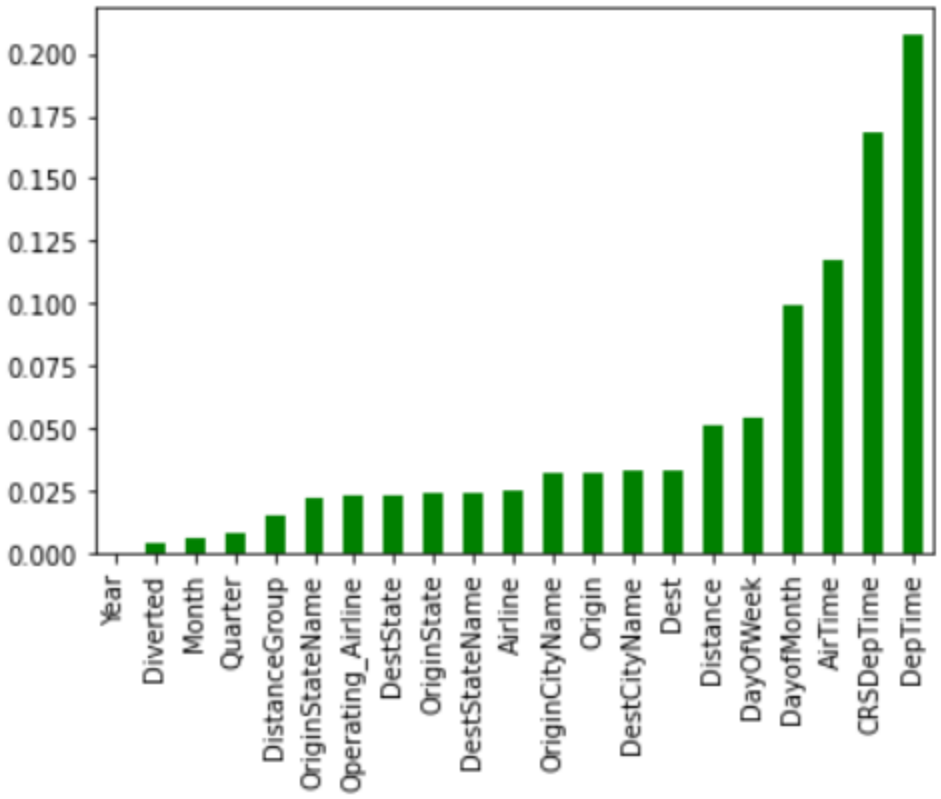
- Data Transformation

This step was introduced to club two variables, DepDelayMinutes and ArrDelayMinutes, and a new variable named Total Delay was created. This was done to reduce the dimensionality of the data and calculate the total Delay contributed by each case.
Based on the above value, a binary variable was created to consider whether a flight was delayed. It is the response variable.

- Label Encoding

The concept of label encoding was implemented to convert non-numerical variables to numerical ones. This approximates the concerned variables to be used for classification.

- **Feature Selection Using Decision Tree:**
  - A decision tree has implicit feature selection during the model-building process. That is, when building the tree, it only does so by splitting on features that cause the greatest increase in node purity, so features that a feature selection method would have eliminated feature that is not used in the model anyway.
  - Here, the features are sorted according to their importance as ranked by the decision tree, and a bar graph is used to visualize the feature importance. This is applied after eliminating some features based on an intuitive interpretation of the business problem and the attributes. It can be observed that Departure Time is the most important feature affecting the response variable.



2) Predicting Delay time

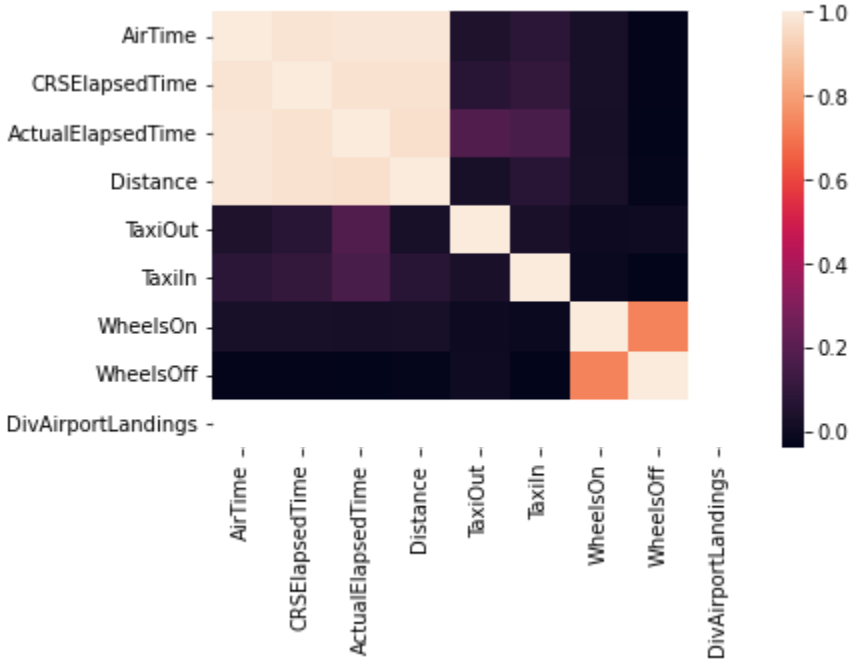
- Column added:
  - The percentage count of delays for each Airline was added for visualization purposes
    Percentage count = (Number of flights delayed per airline / Total number of flights per Airline) * 100

- The data corresponding to canceled flights was removed.
  - **Reason**: The interest lies in predicting delay time. If a flight is canceled, then there is no meaning in studying the delay time.
  - The dimension of data: 1031528, 61

- Handling Missing value:
  - Total number of missing values: 22426, which constitutes 2.1740% of the whole data

- These missing values were removed as imputing these values could be overestimating, affecting the prediction, and dropping such a low percentage of data will not affect the study significantly.

- Handling Multicollinearity:
  - There was a presence of Multicollinearity between the predictors of the Intuitive variable selection approach.



  - Highly correlated variables were removed from the study. As it could impact the study. So, Airtime, CR Elapsed Time, Actual Elapsed Time, and Distance.

- Variable Selection:
  - Approach:
    - Intuitive selection of variables based on business problem
    - Decision tree
  - Reason for the approach:
    - Variable selection is an important step in modeling.
    - The interest was to show how various methods used for variable selection can affect the modeling and how understanding the business problem helps in variable selection.

Variables Used for the study:

1. Classification

   Variables used : DepTime, DestStateName, DistanceGroup, Airline, OriginState, OriginStateName, Operating_Airline, DestState, OriginCityName, CRSDepTime, Origin, DestCityName, Dest, Distance, DayofWeek, DayofMonth, Target.

2. Predicting Delay time

   - **Intuitive approach**
     - **Variable to be predicted**: Dep Delay Minutes
     - **Predictors**: TaxiOut, TaxiIn, Wheels On, Wheels Off, DivAirportLandings

   - **Decision Tree**
     - **Variable to be predicted**: Dep Delay Minutes
     - **Predictors**: Airline, Origin, Destination, AirTime, CRSElapsedTime, ActualElapsedTime, Distance, Month, DayofMonth, TaxiOut, WheelsOff , WheelsOn, TaxiIn, DivAirportLandings.

Model Development:

Below are the models used for the respective objectives:

1. Classification

- **Random Forest**:
  - The Random Forest model here is developed to predict whether a flight is delayed/canceled or on time. It is an ensemble model with decision trees as base learners based on the bagging concept.
  - Bagging is the machine learning technique in which multiple machine learning models are trained on different subsets of the training data to overcome overfitting, improve the model performance by reducing variance and combine predictions from multiple models.

- The Random Forest is an extension of bagging, which uses random sampling to select the features used in each training data subset. It reduces the tendency of decision trees to overfit training data by utilizing the concept of bagging (bootstrap samples) and randomization.

- **Logistic Regression**:
  - Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous, which means two possibilities.
  - It is a special linear regression case where the target variable is categorical. Logistic regression predicts the probability of occurrence of a binary event utilizing a logit function.

- **Naive Bayes Classifier**:
  - Naive Bayes is a statistical classification technique based on Bayes Theorem. The naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. Even if these features are interdependent, these features are still considered independently.
  - Here, the Gaussian Naive Bayes classifier is used, which assumes the likelihood of features to be Gaussian. The parameters are estimated using maximum likelihood.

2. Predicting Delay time

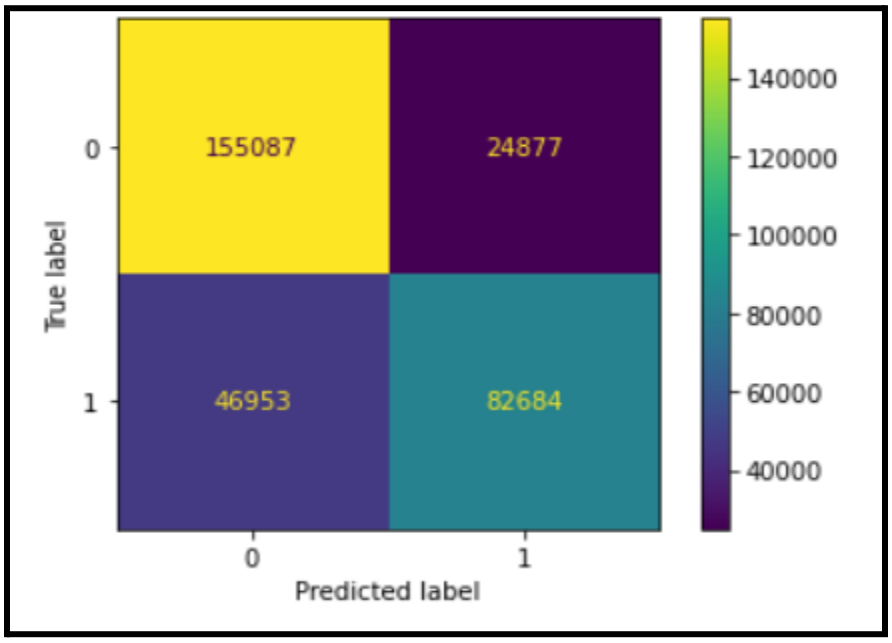
- **Multiple Linear Regression:**
  - Multiple Linear Regression is an extension of Simple Linear regression as it takes more than one predictor variable to predict the response variable. All the variables used should be numerical.
  - Multiple linear regression estimates the relationship between two or more independent variables and one dependent variable.
  - Reason for using Multiple Linear Regression Model: Since all the variables were numeric, the model fitting started with the basic model.

- XGBoost Regression

Results and Interpretation :

1. Classification

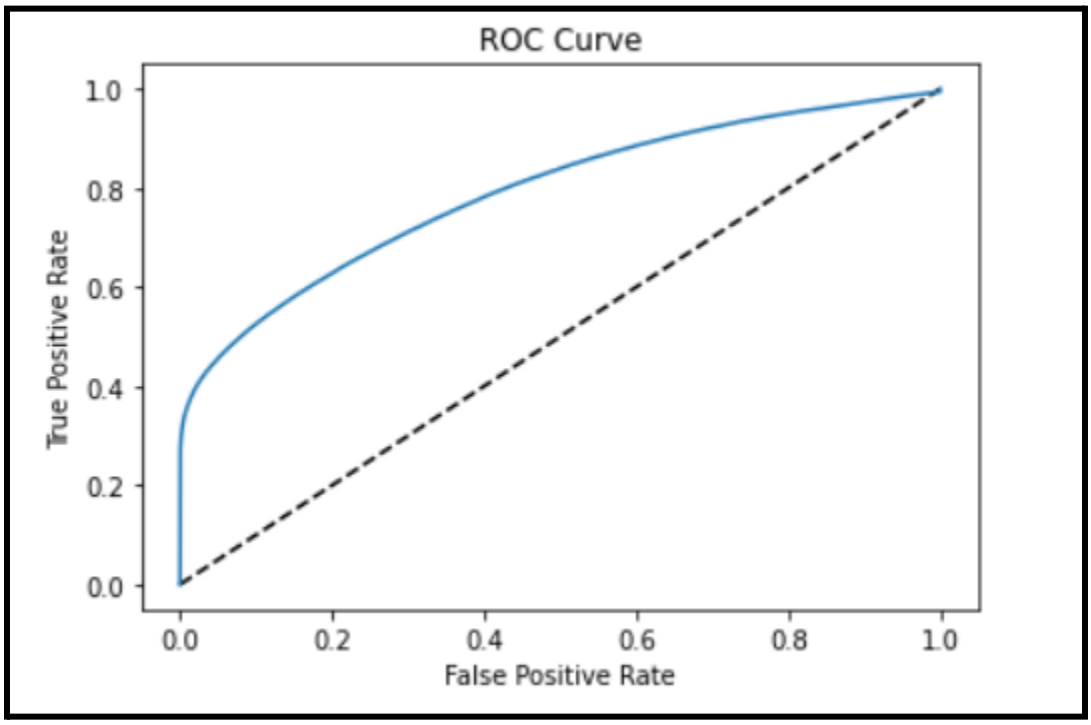
Model Evaluation: Random Forest



Confusion Matrix :
The confusion matrix gives us a glimpse of the precision of our classification technique. The boxes depict True positive, True negative, false positive, and false negative. True positive means the actual value and also the predicted values are coming out the same. True negative is similar to True positive with a difference that symbolizes the opposite value compared to True positive. False positive and False negative means that the actual value is positive, but the model predicts a negative value, and vice versa for false negative.

From the values, it can be concluded that in the case of positive, the model can predict accurate values with few false positives. In the case of negative, the prediction is that there is a larger concentration of false negative value in accordance with True negative. [Precision values are available below]

Model Evaluation: Logistic Regression



Receiver Operating Characteristic (ROC) Curve: An ROC Curve is a graphical representation of the performance of a classification model at all classification thresholds.

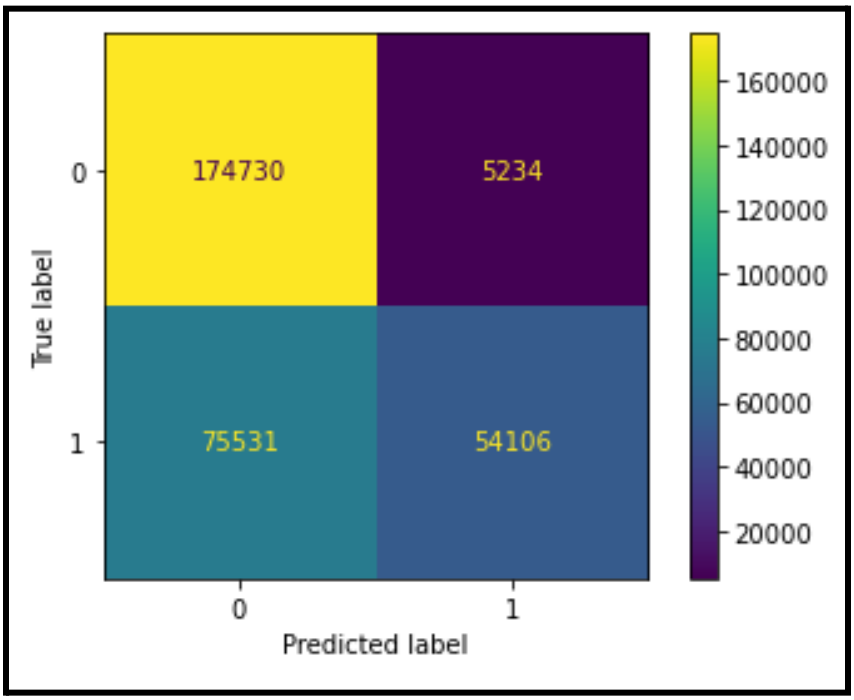
The two parameters in focus are
1) **True Positive Rate:** Canceled or delayed flights that the model predicted correctly
2) **False Positive Rate**: On time flights that the model classified as delayed or canceled

Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.

AUC (Area under the curve) provides an aggregate performance measure across all possible classification thresholds. One way of interpreting AUC is the probability that the model ranks a random positive example more highly than a random negative example. AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

AUC SCORE = 0.6014

Confusion matrix



From the above graph and understanding of the Confusion matrix, it can be concluded that the model can predict values accurately and precisely in the maximum number of cases with lesser false cases.

A Comparison of the Classification Models

Classification Report: Random Forest

	precision	recall	f1-score	support
0	0.77	0.86	0.81	179964
1	0.77	0.64	0.70	129637
accuracy			0.77	309601
macro avg	0.77	0.75	0.75	309601
weighted avg	0.77	0.77	0.76	309601

Classification Report: Logistic Regression

	precision	recall	f1-score	support
0	0.70	0.97	0.81	179964
1	0.91	0.42	0.57	129637
accuracy			0.74	309601
macro avg	0.80	0.69	0.69	309601
weighted avg	0.79	0.74	0.71	309601

Classification Report: Gaussian Naive Bayes

	precision	recall	f1-score	support
0	0.62	0.76	0.69	179964
1	0.52	0.36	0.42	129637
accuracy			0.59	309601
macro avg	0.57	0.56	0.56	309601
weighted avg	0.58	0.59	0.58	309601

Interpretation:

Model	Precision		Recall		F1-Score		Accuracy
Random Forest	0.77	0.77	0.86	0.64	0.81	0.70	77%
Logistic Regression	0.70	0.91	0.97	0.42	0.81	0.57	74%
Gaussian Naive Bayes	0.62	0.52	0.76	0.36	0.69	0.42	59%

The F1 score is a reliable metric to evaluate and compare the performance of multiple classification models. According to the classification reports generated above, the random forest classifier performs fairly better than the logistic regression and Gaussian Naive Bayes classifier in terms of F1 score and accuracy.

2. Predicting Delay time

A Comparison of the prediction Models

- Intuitive variable selection

Model	RMSE
MLR	**42.5**
Random Forest	**43.72**
XGBoost Regression	**41.54**

- Variable selection using decision tree

Model	RMSE
Random Forest	**39.55**
XGBoost Regression	**41**

When intuitive variables are selected, XGBoost Regression provides the best prediction for the given data. Random Forest Regression is the best choice when decision trees are used for selecting the variables for building the prediction model.

Conclusion:

The methods used in this study for classification were Random Forest, Logistic Regression, and Gaussian Naive Bayes. Out of all these methods, Random forest had the highest accuracy of 77%.

Based on the feature selection by the Decision tree, various variables come into the light. First and foremost is Departure time. This suggests that departure time is an important factor contributing to delay. Air time, i.e., the time a flight is in the air also plays an important role in contributing to delay. The time of month and day of the week also are critical ensembles to delay metrics.

Overall the conclusion that can be drawn from the features is that departure time, air time and day of week affect the delay in a greater context and in order to reduce the delay few suggestions can be made. First is departure time must be strict and accurately followed. day of the week and time of the month can be an essential factor for the delay, it can be attributed to the fact that, for example, on weekends an airport experiences a significant increase in footfall and increase in demand. So proper arrangements must be made during peak days to reduce delays.

The method used for variable selection plays an important role. When the Decision tree method was applied to the study to choose the variable, many variables were selected simultaneously. It did not improve the prediction of the model significantly.

The delay of the departure is affected by several controllable and uncontrollable factors. The uncontrollable factors include Diverted Airport Landing, which occurs due to various uncontrollable occurrences.

The Airlines will be able to reduce the departure delay only if they concentrate on controllable factors like Taxi In, Taxi out, Wheels on, and Wheels off and by eliminating the difference between CRSE and Actual Elapsed time.

Prescription:

- In order to reduce the delay in departure the Airline has to optimize the Taxi out and Taxi in time. This can be done by timely maintenance and by increasing the frequency of maintenance of the aircraft.

- The departure time is an important factor which significantly affects the response variable in the classification model, hence the airline company can optimize their schedule and departure times such that delays or flight cancellations do not occur.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv(r"C:\Users\kinne\Downloads\Flight_status.csv")
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 61 columns):
 #   Column                                  Non-Null Count    Dtype
---  ------                                  --------------    -----
 0   FlightDate                              1048575 non-null  object
 1   Airline                                 1048575 non-null  object
 2   Origin                                  1048575 non-null  object
 3   Dest                                    1048575 non-null  object
 4   Cancelled                               1048575 non-null  bool
 5   Diverted                                1048575 non-null  bool
 6   CRSDepTime                              1048575 non-null  int64
 7   DepTime                                 1032002 non-null  float64
 8   DepDelayMinutes                         1031514 non-null  float64
 9   DepDelay                                1031514 non-null  float64
 10  ArrTime                                 1031060 non-null  float64
 11  ArrDelayMinutes                         1029014 non-null  float64
 12  AirTime                                 1028355 non-null  float64
 13  CRSElapsedTime                          1048575 non-null  int64
 14  ActualElapsedTime                       1029216 non-null  float64
 15  Distance                                1048575 non-null  int64
 16  Year                                    1048575 non-null  int64
 17  Quarter                                 1048575 non-null  int64
 18  Month                                   1048575 non-null  int64
 19  DayofMonth                              1048575 non-null  int64
 20  DayOfWeek                               1048575 non-null  int64
 21  Marketing_Airline_Network               1048575 non-null  object
 22  Operated_or_Branded_Code_Share_Partners 1048575 non-null  object
 23  DOT_ID_Marketing_Airline                1048575 non-null  int64
 24  IATA_Code_Marketing_Airline             1048575 non-null  object
 25  Flight_Number_Marketing_Airline         1048575 non-null  int64
 26  Operating_Airline                       1048575 non-null  object
 27  DOT_ID_Operating_Airline                1048575 non-null  int64
 28  IATA_Code_Operating_Airline             1048575 non-null  object
 29  Tail_Number                             1045824 non-null  object
 30  Flight_Number_Operating_Airline         1048575 non-null  int64
 31  OriginAirportID                         1048575 non-null  int64
 32  OriginAirportSeqID                      1048575 non-null  int64
 33  OriginCityMarketID                      1048575 non-null  int64
 34  OriginCityName                          1048575 non-null  object
 35  OriginState                             1048575 non-null  object
 36  OriginStateFips                         1048575 non-null  int64
 37  OriginStateName                         1048575 non-null  object
 38  OriginWac                               1048575 non-null  int64
 39  DestAirportID                           1048575 non-null  int64
 40  DestAirportSeqID                        1048575 non-null  int64
 41  DestCityMarketID                        1048575 non-null  int64
 42  DestCityName                            1048575 non-null  object
 43  DestState                               1048575 non-null  object
 44  DestStateFips                           1048575 non-null  int64
 45  DestStateName                           1048575 non-null  object
 46  DestWac                                 1048575 non-null  int64
 47  DepDel15                                1031514 non-null  float64
 48  DepartureDelayGroups                    1031514 non-null  float64
```

```
 49  DepTimeBlk                                1048575 non-null  object
 50  TaxiOut                                   1030819 non-null  float64
 51  WheelsOff                                 1030828 non-null  float64
 52  WheelsOn                                  1030208 non-null  float64
 53  TaxiIn                                    1030199 non-null  float64
 54  CRSArrTime                                1048575 non-null  int64
 55  ArrDelay                                  1029014 non-null  float64
 56  ArrDel15                                  1029014 non-null  float64
 57  ArrivalDelayGroups                        1029014 non-null  float64
 58  ArrTimeBlk                                1048575 non-null  object
 59  DistanceGroup                             1048575 non-null  int64
 60  DivAirportLandings                        1048575 non-null  int64
dtypes: bool(2), float64(16), int64(25), object(18)
memory usage: 474.0+ MB
```

In [4]:
```
df.shape
```

Out[4]:
```
(1048575, 61)
```

In [5]:
```
df['Cancelled'].value_counts()
```

Out[5]:
```
False    1031528
True       17047
Name: Cancelled, dtype: int64
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['FlightDate', 'Airline', 'Origin', 'Dest', 'Cancelled', 'Diverted',
       'CRSDepTime', 'DepTime', 'DepDelayMinutes', 'DepDelay', 'ArrTime',
       'ArrDelayMinutes', 'AirTime', 'CRSElapsedTime', 'ActualElapsedTime',
       'Distance', 'Year', 'Quarter', 'Month', 'DayofMonth', 'DayOfWeek',
       'Marketing_Airline_Network', 'Operated_or_Branded_Code_Share_Partners
',
       'DOT_ID_Marketing_Airline', 'IATA_Code_Marketing_Airline',
       'Flight_Number_Marketing_Airline', 'Operating_Airline',
       'DOT_ID_Operating_Airline', 'IATA_Code_Operating_Airline',
       'Tail_Number', 'Flight_Number_Operating_Airline', 'OriginAirportID',
       'OriginAirportSeqID', 'OriginCityMarketID', 'OriginCityName',
       'OriginState', 'OriginStateFips', 'OriginStateName', 'OriginWac',
       'DestAirportID', 'DestAirportSeqID', 'DestCityMarketID', 'DestCityName
',
       'DestState', 'DestStateFips', 'DestStateName', 'DestWac', 'DepDel15',
       'DepartureDelayGroups', 'DepTimeBlk', 'TaxiOut', 'WheelsOff',
       'WheelsOn', 'TaxiIn', 'CRSArrTime', 'ArrDelay', 'ArrDel15',
       'ArrivalDelayGroups', 'ArrTimeBlk', 'DistanceGroup',
       'DivAirportLandings'],
      dtype='object')
```

In [7]:
```
'FlightDate',
'Airline',
'Origin',
'Dest',
'Cancelled',
'AirTime',
'Distance',
'Year',
'Quarter',
'Month',
'DayofMonth',
'DayOfWeek',
'Operating_Airline',
'OriginCityName',
'OriginState',
'OriginStateName',
'DestCityName',
'DestState',
'DestStateName',
'DistanceGroup'
```

Out[7]: 'DistanceGroup'

In [8]:
```
df_mod = df[['Airline','Origin','Dest','Cancelled','Diverted','CRSDepTime', 'D
             'ArrDelayMinutes','AirTime','CRSElapsedTime', 'ActualElapsedTime'
              'Operating_Airline','OriginCityName','OriginState','OriginStateN
              'DestState','DestStateName','DepDel15','DepartureDelayGroups', '
              'TaxiIn', 'CRSArrTime', 'ArrDelay', 'ArrDel15',
              'ArrivalDelayGroups', 'ArrTimeBlk','DistanceGroup']]
```

In [9]:
```
df_mod.head()
```

Out[9]:

	Airline	Origin	Dest	Cancelled	Diverted	CRSDepTime	DepTime	DepDelayMinutes	DepDe
0	Endeavor Air Inc.	ABY	ATL	False	False	1202	1157.0	0.0	-
1	Endeavor Air Inc.	ABY	ATL	False	False	1202	1157.0	0.0	-
2	Endeavor Air Inc.	ABY	ATL	False	False	1202	1153.0	0.0	-
3	Endeavor Air Inc.	ABY	ATL	False	False	1202	1150.0	0.0	-1
4	Endeavor Air Inc.	ABY	ATL	False	False	1400	1355.0	0.0	-

5 rows × 40 columns

In [10]:
```
df_mod.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 40 columns):
 #   Column                Non-Null Count    Dtype  
---  ------                --------------    -----  
 0   Airline               1048575 non-null  object 
 1   Origin                1048575 non-null  object 
 2   Dest                  1048575 non-null  object 
 3   Cancelled             1048575 non-null  bool   
 4   Diverted              1048575 non-null  bool   
 5   CRSDepTime            1048575 non-null  int64  
 6   DepTime               1032002 non-null  float64
 7   DepDelayMinutes       1031514 non-null  float64
 8   DepDelay              1031514 non-null  float64
 9   ArrTime               1031060 non-null  float64
 10  ArrDelayMinutes       1029014 non-null  float64
 11  AirTime               1028355 non-null  float64
 12  CRSElapsedTime        1048575 non-null  int64  
 13  ActualElapsedTime     1029216 non-null  float64
 14  Distance              1048575 non-null  int64  
 15  Year                  1048575 non-null  int64  
 16  Quarter               1048575 non-null  int64  
 17  Month                 1048575 non-null  int64  
 18  DayofMonth            1048575 non-null  int64  
 19  DayOfWeek             1048575 non-null  int64  
 20  Operating_Airline     1048575 non-null  object 
 21  OriginCityName        1048575 non-null  object 
 22  OriginState           1048575 non-null  object 
 23  OriginStateName       1048575 non-null  object 
 24  DestCityName          1048575 non-null  object 
 25  DestState             1048575 non-null  object 
 26  DestStateName         1048575 non-null  object 
 27  DepDel15              1031514 non-null  float64
 28  DepartureDelayGroups  1031514 non-null  float64
 29  DepTimeBlk            1048575 non-null  object 
 30  TaxiOut               1030819 non-null  float64
 31  WheelsOff             1030828 non-null  float64
 32  WheelsOn              1030208 non-null  float64
 33  TaxiIn                1030199 non-null  float64
 34  CRSArrTime            1048575 non-null  int64  
 35  ArrDelay              1029014 non-null  float64
 36  ArrDel15              1029014 non-null  float64
 37  ArrivalDelayGroups    1029014 non-null  float64
 38  ArrTimeBlk            1048575 non-null  object 
 39  DistanceGroup         1048575 non-null  int64  
dtypes: bool(2), float64(16), int64(10), object(12)
memory usage: 306.0+ MB
```

In [11]:
```
df_mod['Cancelled'].value_counts()
```

Out[11]:
```
False    1031528
True       17047
Name: Cancelled, dtype: int64
```

In [12]:
```
df_mod.isnull().sum().sort_values(ascending=False)/len(df)
```

Out[12]:
```
AirTime                 0.019283
ArrDelayMinutes         0.018655
ArrivalDelayGroups      0.018655
ArrDel15                0.018655
ArrDelay                0.018655
ActualElapsedTime       0.018462
TaxiIn                  0.017525
WheelsOn                0.017516
TaxiOut                 0.016933
WheelsOff               0.016925
ArrTime                 0.016704
DepDelay                0.016271
DepDel15                0.016271
DepartureDelayGroups    0.016271
DepDelayMinutes         0.016271
DepTime                 0.015805
DepTimeBlk              0.000000
DestStateName           0.000000
Airline                 0.000000
CRSArrTime              0.000000
DestCityName            0.000000
ArrTimeBlk              0.000000
DestState               0.000000
Operating_Airline       0.000000
OriginStateName         0.000000
OriginState             0.000000
OriginCityName          0.000000
Origin                  0.000000
DayOfWeek               0.000000
DayofMonth              0.000000
Month                   0.000000
Quarter                 0.000000
Year                    0.000000
Distance                0.000000
CRSElapsedTime          0.000000
CRSDepTime              0.000000
Diverted                0.000000
Cancelled               0.000000
Dest                    0.000000
DistanceGroup           0.000000
dtype: float64
```

In [13]:
```
missing_cols_1 = ['AirTime','ArrDelayMinutes','ArrDel15','ArrDelay','ActualEla
                  'TaxiOut','WheelsOff','ArrTime','DepDelay','DepDel15','DepDelay


for i in missing_cols_1:
    df_mod.loc[df_mod.loc[:,i].isnull(),i]=df_mod.loc[:,i].mean()
```

```
C:\ANACONDA\lib\site-packages\pandas\core\indexing.py:1817: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  self._setitem_single_column(loc, value, pi)
```

In [14]:
```
df_mod['Total_Delay'] = df_mod['DepDelayMinutes'] + df_mod['ArrDelayMinutes']
```

```
C:\Users\kinne\AppData\Local\Temp/ipykernel_21932/3026156306.py:1: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df_mod['Total_Delay'] = df_mod['DepDelayMinutes'] + df_mod['ArrDelayMinutes
']
```

In [15]:
```
df_mod['Delay'] = np.where(df_mod['Total_Delay'] >  0, True, False)
```

```
C:\Users\kinne\AppData\Local\Temp/ipykernel_21932/3857171070.py:1: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df_mod['Delay'] = np.where(df_mod['Total_Delay'] >  0, True, False)
```

In [16]:
```
df_mod['Target'] = np.where((df_mod['Delay'] == True) | (df_mod['Cancelled'] =
```

```
C:\Users\kinne\AppData\Local\Temp/ipykernel_21932/53465250.py:1: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df_mod['Target'] = np.where((df_mod['Delay'] == True) | (df_mod['Cancelled
'] == True), 1, 0)
```

In [17]:
```
df_mod['Target'].value_counts()
```

Out[17]:
```
0    599880
1    448695
Name: Target, dtype: int64
```

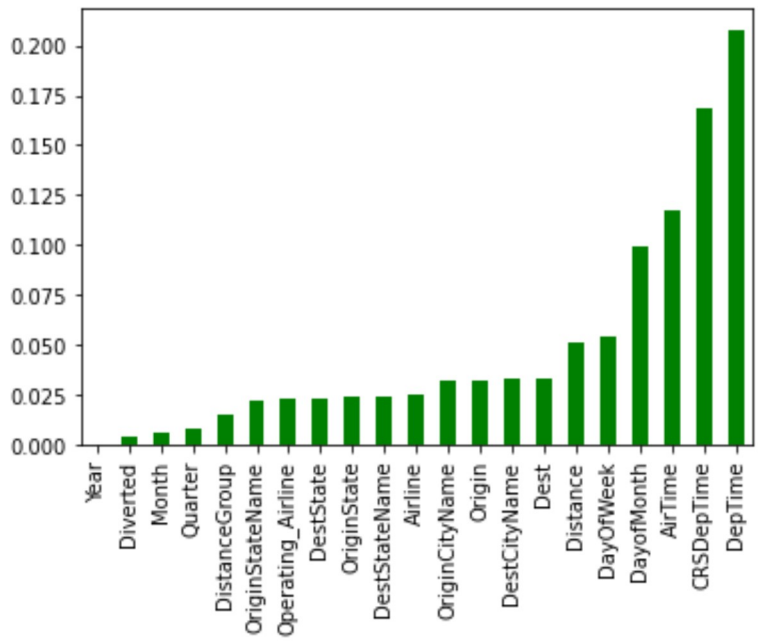
In [18]:
```
df_mod = df_mod.drop(['ArrTime','CRSElapsedTime', 'ActualElapsedTime',
             'DepTimeBlk', 'TaxiOut', 'WheelsOff','WheelsOn', 'TaxiIn', 'CRSAr
             'ArrTimeBlk','Total_Delay', 'ArrivalDelayGroups','DepartureDelayG
             'DepDelay','DepDel15','DepDelayMinutes','ArrDelayMinutes','ArrDel
```

In [19]:
```
from sklearn import preprocessing

list2 = ['Airline','Origin','Dest','Operating_Airline','OriginCityName','Origi
          'DestState','DestStateName','DistanceGroup']
label_encoder = preprocessing.LabelEncoder()

for column in list2:
    df_mod[column] = label_encoder.fit_transform(df_mod[column])
```

In [20]:
```
from sklearn.model_selection import train_test_split

X = df_mod.drop('Target', axis=1).values
y = df_mod['Target'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, strat
```

In [21]:
```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(criterion='gini',splitter='best',min_samples_split

dt.fit(X_train, y_train)

y_pred = dt.predict(X_test)
```

In [22]:
```
X = df_mod.drop('Target', axis=1)
y = df_mod['Target']

feature_importance = pd.Series(dt.feature_importances_, index = X.columns)

sorted_importance = feature_importance.sort_values()

sorted_importance.plot(kind='bar',color='green');

plt.show()
```



In [23]:
```
df_mod.columns
```

Out[23]:
```
Index(['Airline', 'Origin', 'Dest', 'Diverted', 'CRSDepTime', 'DepTime',
       'AirTime', 'Distance', 'Year', 'Quarter', 'Month', 'DayofMonth',
       'DayOfWeek', 'Operating_Airline', 'OriginCityName', 'OriginState',
       'OriginStateName', 'DestCityName', 'DestState', 'DestStateName',
       'DistanceGroup', 'Target'],
      dtype='object')
```

In [24]:
```
df_opt = df[['DepTime','CRSDepTime','DayofMonth','DayOfWeek','Distance','Dest'
             'DestStateName','DestState','Operating_Airline','OriginStateName'
```

In [25]:
```
tseries = df_mod['Target'].squeeze()
```

In [26]:
```
df_opt['Target'] = tseries.values
```

```
C:\Users\kinne\AppData\Local\Temp/ipykernel_21932/3099269131.py:1: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df_opt['Target'] = tseries.values
```

In [27]:
```
df_opt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 17 columns):
 #   Column             Non-Null Count    Dtype
---  ------             --------------    -----
 0   DepTime            1032002 non-null  float64
 1   CRSDepTime         1048575 non-null  int64
 2   DayofMonth         1048575 non-null  int64
 3   DayOfWeek          1048575 non-null  int64
 4   Distance           1048575 non-null  int64
 5   Dest               1048575 non-null  object
 6   DestCityName       1048575 non-null  object
 7   Origin             1048575 non-null  object
 8   OriginCityName     1048575 non-null  object
 9   DestStateName      1048575 non-null  object
 10  DestState          1048575 non-null  object
 11  Operating_Airline  1048575 non-null  object
 12  OriginStateName    1048575 non-null  object
 13  OriginState        1048575 non-null  object
 14  Airline            1048575 non-null  object
 15  DistanceGroup      1048575 non-null  int64
 16  Target             1048575 non-null  int32
dtypes: float64(1), int32(1), int64(5), object(10)
memory usage: 132.0+ MB
```

In [28]:
```
from sklearn import preprocessing

list3 = ['Dest','DestCityName','Origin','OriginCityName',
             'DestStateName','DestState','Operating_Airline','OriginStateName'
label_encoder = preprocessing.LabelEncoder()

for column in list2:
    df_opt[column] = label_encoder.fit_transform(df_opt[column])
```

```
C:\Users\kinne\AppData\Local\Temp/ipykernel_21932/2241169188.py:8: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df_opt[column] = label_encoder.fit_transform(df_opt[column])
```

In [29]:
```
df_opt.isnull().sum().sort_values(ascending=False)/len(df)
```

Out[29]:
```
DepTime              0.015805
DestStateName        0.000000
DistanceGroup        0.000000
Airline              0.000000
OriginState          0.000000
OriginStateName      0.000000
Operating_Airline    0.000000
DestState            0.000000
OriginCityName       0.000000
CRSDepTime           0.000000
Origin               0.000000
DestCityName         0.000000
Dest                 0.000000
Distance             0.000000
DayOfWeek            0.000000
DayofMonth           0.000000
Target               0.000000
dtype: float64
```

In [30]:
```
df_opt.dropna(subset = ["DepTime"], inplace =True )
```

```
C:\ANACONDA\lib\site-packages\pandas\util\_decorators.py:311: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  return func(*args, **kwargs)
```

In [31]:
```
X = df_opt.drop('Target', axis=1).values
y = df_opt['Target'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, strat
```

In [32]:
```
# Random Forest
from sklearn.model_selection import cross_val_score, KFold
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1)

kf = KFold(n_splits=5, random_state=29, shuffle=True)
cv_results = cross_val_score(rf, X_train, y_train, cv=kf, error_score='raise')

mean_accuracy = cv_results.mean()
median_accuracy = np.median(cv_results)
```

In [33]:
```
print("Mean Ensemble Accuracy:{}".format(mean_accuracy))
print("Median Ensemble Accuracy:{}".format(median_accuracy))
```

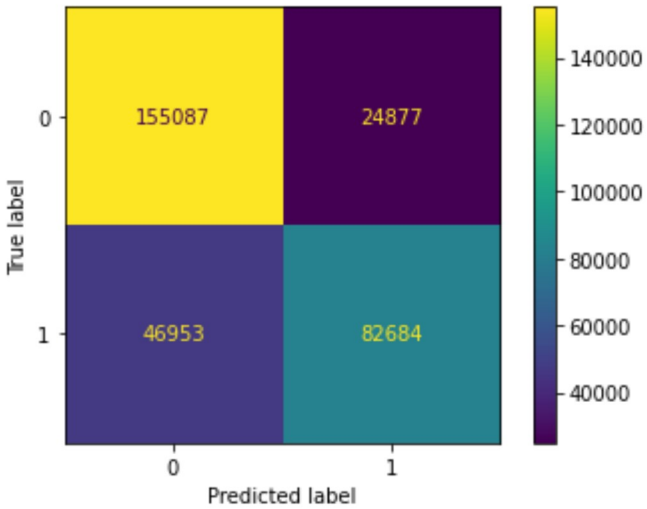
```
Mean Ensemble Accuracy:0.7580028283616411
Median Ensemble Accuracy:0.7574473975636766
```

In [34]:
```
rf.fit(X_train, y_train)
```

Out[34]:
```
RandomForestClassifier(n_jobs=-1, random_state=42)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [35]:
```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(rf, X_test, y_test)
plt.show()
```

```
C:\ANACONDA\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning:
Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix
` is deprecated in 1.0 and will be removed in 1.2. Use one of the class metho
ds: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_es
timator.
  warnings.warn(msg, category=FutureWarning)
```



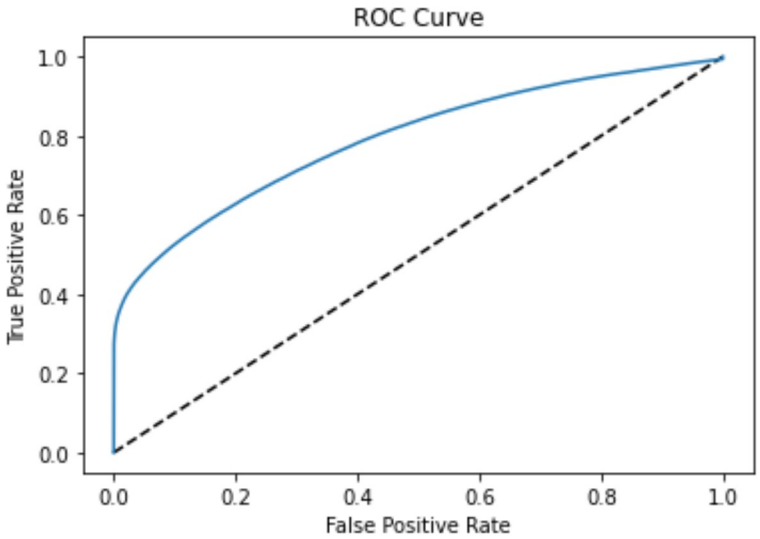
In [36]:
```
from sklearn.metrics import classification_report

y_pred = rf.predict(X_test)
y_pred_probs = rf.predict_proba(X_test)[:, 1]

# Calculate the classification report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.77      0.86      0.81    179964
           1       0.77      0.64      0.70    129637

    accuracy                           0.77    309601
   macro avg       0.77      0.75      0.75    309601
weighted avg       0.77      0.77      0.76    309601
```

In [37]:
```
# Logistic Regression

# Import LogisticRegression
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Instantiate the model
logreg = LogisticRegression()

# Fit the model
logreg.fit(X_train, y_train)

#Predict the labels
y_pred = logreg.predict(X_test)

# Predict probabilities
y_pred_probs = logreg.predict_proba(X_test)[:, 1]
```

```
C:\ANACONDA\lib\site-packages\sklearn\linear_model\_logistic.py:444: Converge
nceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

In [38]:
```
# Import roc_curve
from sklearn.metrics import roc_curve

# Generate ROC curve values: fpr, tpr, thresholds
fpr, tpr, thresholds = roc_curve(y_test, y_pred_probs)

plt.plot([0, 1], [0, 1], 'k--')

# Plot tpr against fpr
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
```

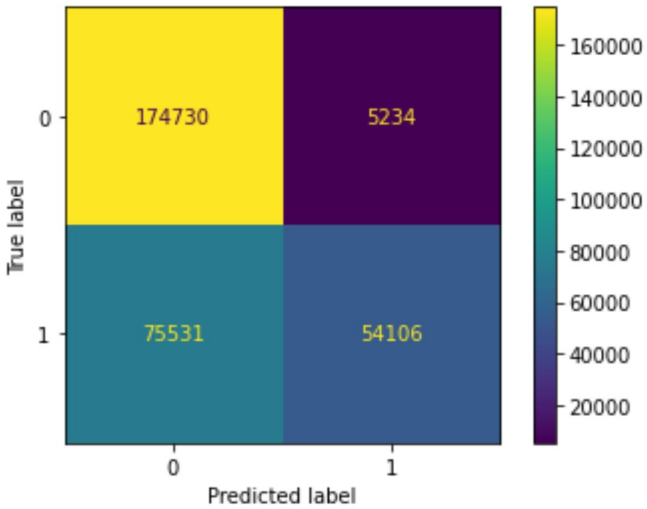


In [45]:
```
from sklearn.metrics import roc_auc_score

auc = roc_auc_score(y_test, y_pred_probs)
print(auc)
```

```
0.6014638997791578
```

In [39]:
```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(logreg, X_test, y_test)
plt.show()
```

```
C:\ANACONDA\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning:
Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix
` is deprecated in 1.0 and will be removed in 1.2. Use one of the class metho
ds: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_es
timator.
  warnings.warn(msg, category=FutureWarning)
```



In [40]:
```
from sklearn.metrics import classification_report

y_pred = logreg.predict(X_test)
y_pred_probs = logreg.predict_proba(X_test)[:, 1]

# Calculate the classification report
print(classification_report(y_test, y_pred))
```

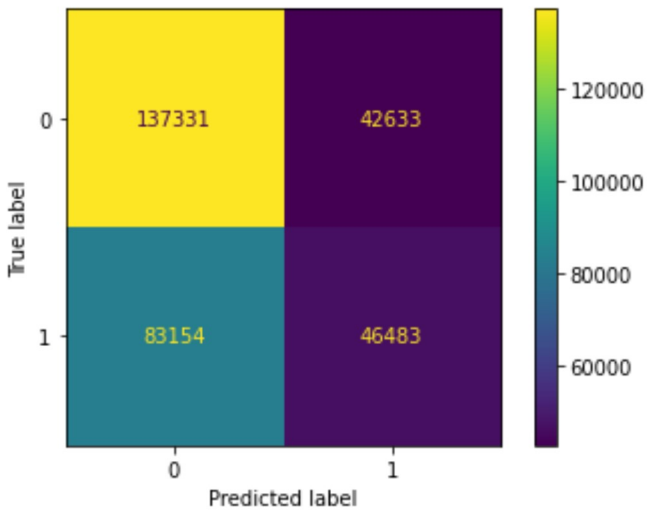
```
              precision    recall  f1-score   support

           0       0.70      0.97      0.81    179964
           1       0.91      0.42      0.57    129637

    accuracy                           0.74    309601
   macro avg       0.80      0.69      0.69    309601
weighted avg       0.79      0.74      0.71    309601
```

In [41]:
```
# Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()
y_pred = gnb.fit(X_train, y_train).predict(X_test)
print("Number of mislabeled points out of a total %d points : %d"
      % (X_test.shape[0], (y_test != y_pred).sum()))
```

```
Number of mislabeled points out of a total 309601 points : 125787
```

In [42]:
```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(gnb, X_test, y_test)
plt.show()
```

```
C:\ANACONDA\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning:
Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix
` is deprecated in 1.0 and will be removed in 1.2. Use one of the class metho
ds: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_es
timator.
  warnings.warn(msg, category=FutureWarning)
```



In [43]:
```
from sklearn.metrics import classification_report

y_pred = gnb.predict(X_test)
y_pred_probs = gnb.predict_proba(X_test)[:, 1]

# Calculate the classification report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.62      0.76      0.69    179964
           1       0.52      0.36      0.42    129637

    accuracy                           0.59    309601
   macro avg       0.57      0.56      0.56    309601
weighted avg       0.58      0.59      0.58    309601
```

In [ ]:

In [1]:
```
#Data loading
import pandas as pd
import numpy as np
import seaborn as sns
```

In [2]:
```
fli=pd.read_csv(r"C:\Users\amrit\OneDrive\Desktop\sem 3\SLDM\Mini Project\Flight_
```

In [3]:
```
f=fli.copy()
f.head()
```

Out[3]:

	FlightDate	Airline	Origin	Dest	Cancelled	Diverted	CRSDepTime	DepTime	DepDelayMinute
0	23-01-2018	Endeavor Air Inc.	ABY	ATL	False	False	1202	1157.0	0
1	24-01-2018	Endeavor Air Inc.	ABY	ATL	False	False	1202	1157.0	0
2	25-01-2018	Endeavor Air Inc.	ABY	ATL	False	False	1202	1153.0	0
3	26-01-2018	Endeavor Air Inc.	ABY	ATL	False	False	1202	1150.0	0
4	27-01-2018	Endeavor Air Inc.	ABY	ATL	False	False	1400	1355.0	0

5 rows × 61 columns

In [4]:
```
f.shape
```

Out[4]:
```
(1048575, 61)
```

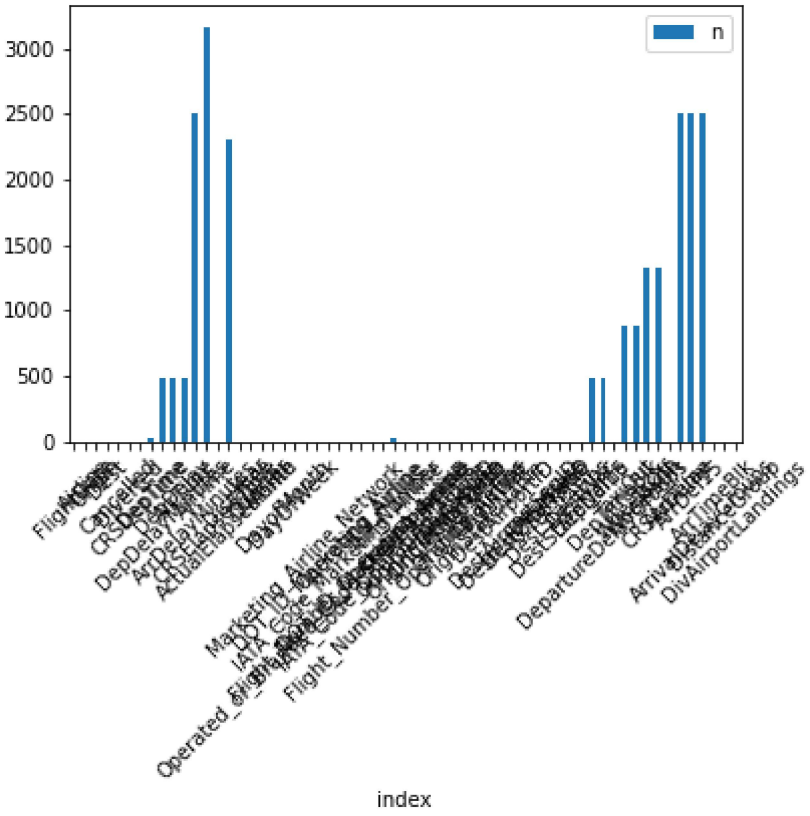
In [5]:
```
grouped = f.groupby(f.Cancelled)
flight= grouped.get_group(False)
```

In [6]:
```
flight.shape
```

Out[6]:
```
(1031528, 61)
```

In [7]:
```
#Seeing the na values in in the
flight.isna().sum().reset_index(name="n").plot.bar(x='index', y='n', rot=45)
```

Out[7]: <AxesSubplot:xlabel='index'>



In [8]:
```
k=np.where(flight.isna().sum())
print(k)
print(flight.isna().sum().sum())#22426- 2.1740563% of data are missing
flight.iloc[:,[7,8,9,10,11,12,14,29,47,48,50,51,52,53,55,56,57]].describe()
```

```
(array([ 7,  8,  9, 10, 11, 12, 14, 29, 47, 48, 50, 51, 52, 53, 55, 56, 57],
      dtype=int64),)
22426
```

Out[8]:

	DepTime	DepDelayMinutes	DepDelay	ArrTime	ArrDelayMinutes	AirTim
count	1.031509e+06	1.031038e+06	1.031038e+06	1.031051e+06	1.029014e+06	1.028355e+0
mean	1.331116e+03	1.181459e+01	8.235016e+00	1.477454e+03	1.195543e+01	1.084320e+0
std	4.948459e+02	4.279688e+01	4.399000e+01	5.218854e+02	4.253059e+01	6.990448e+0
min	1.000000e+00	0.000000e+00	-1.280000e+03	1.000000e+00	0.000000e+00	-1.244000e+0
25%	9.190000e+02	0.000000e+00	-6.000000e+00	1.057000e+03	0.000000e+00	5.800000e+0
50%	1.327000e+03	0.000000e+00	-3.000000e+00	1.510000e+03	0.000000e+00	8.900000e+0
75%	1.737000e+03	5.000000e+00	5.000000e+00	1.914000e+03	6.000000e+00	1.370000e+0
max	2.400000e+03	2.109000e+03	2.109000e+03	2.400000e+03	2.153000e+03	6.830000e+0

In [9]:
```
flight=flight.dropna()#Droping the NA values as it constitutes only 2.17% of data
```

In [10]:
```
print(flight.shape)
```

```
(1027677, 61)
```

In [61]:
```
################################## Intutive Variable Selection ####################
```

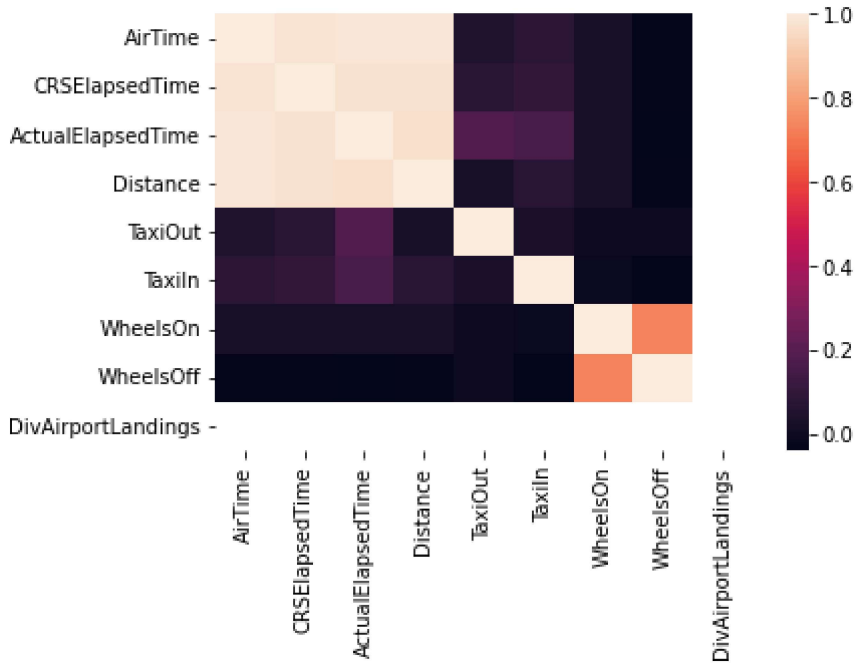
In [62]:
```
f_int=flight[['AirTime','CRSElapsedTime','ActualElapsedTime','Distance','TaxiOut'
```

In [64]:
```
f_int.head()
```

Out[64]:

	AirTime	CRSElapsedTime	ActualElapsedTime	Distance	TaxiOut	TaxiIn	WheelsOn	WheelsOff
0	38.0	62	59.0	145	14.0	7.0	1249.0	1211.0
1	36.0	62	61.0	145	13.0	12.0	1246.0	1210.0
2	40.0	62	69.0	145	18.0	11.0	1251.0	1211.0
3	35.0	62	63.0	145	17.0	11.0	1242.0	1207.0
4	36.0	60	64.0	145	17.0	11.0	1448.0	1412.0

In [97]:
```
y=f_int.iloc[:,9]
X=f_int.iloc[:,0:9]

c=X.corr()
sns.heatmap(c);
```



In [82]:
```
y=f_int.iloc[:,9]
X=f_int.iloc[:,4:9]
```

In [84]:
```
#splitting 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=0.3, random_s
```

In [86]:
```
######################    Multiple Linear Regression    ######################
```

In [87]:
```
from sklearn import linear_model
regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)
```

Out[87]:
```
LinearRegression()
```

In [88]:
```
pred = regr.predict(X_test)
```

In [90]:
```
from sklearn.metrics import mean_squared_error
mse=mean_squared_error(y_test,pred)
import math
rmse=math.sqrt(mse)
rmse
```

Out[90]:
```
42.49610771479272
```

In [91]:
```
######################    XGBoost Regression   ######################

from sklearn.ensemble import GradientBoostingRegressor

gbr = GradientBoostingRegressor(n_estimators = 100, random_state = 0)

gbr.fit(X_train, y_train)

pred = gbr.predict(X_test)
```

In [92]:
```
from sklearn.metrics import mean_squared_error
mse=mean_squared_error(y_test,pred)
import math
rmse=math.sqrt(mse)
rmse
```

Out[92]:
```
41.53603366936902
```

In [94]:
```
######################   Random Forest ############################


from sklearn.ensemble import RandomForestRegressor

 # create regressor object
regressor = RandomForestRegressor(n_estimators = 100, random_state = 0)
regressor.fit(X_train, y_train)
pred1=regressor.predict(X_test)
```

In [95]:
```
from sklearn.metrics import mean_squared_error
mse=mean_squared_error(y_test,pred1)
import math
rmse=math.sqrt(mse)
rmse
```

Out[95]:
```
43.719350479210846
```

In [ ]:
```
############## Variable Selection ############
```

In [134]:
```
flight_used = flight[['Airline','Origin','Dest','DepDelay',
             'AirTime','CRSElapsedTime', 'ActualElapsedTime','Distance','Month',
              'TaxiIn', 'ArrDelay','DivAirportLandings','DepDelayMinutes']]
```

In [135]:
```
flight_used.shape
```

Out[135]:
```
(1027677, 17)
```

In [136]:
```
flight_used.isnull().sum()
```

Out[136]:
```
Airline               0
Origin                0
Dest                  0
DepDelay              0
AirTime               0
CRSElapsedTime        0
ActualElapsedTime     0
Distance              0
Month                 0
DayofMonth            0
TaxiOut               0
WheelsOff             0
WheelsOn              0
TaxiIn                0
ArrDelay              0
DivAirportLandings    0
DepDelayMinutes       0
dtype: int64
```

In [138]:
```
from sklearn import preprocessing

list2 = ['Airline','Origin','Dest']

label_encoder = preprocessing.LabelEncoder()

for column in list2:
    flight_used[column] = label_encoder.fit_transform(flight_used[column])
```

```
C:\Users\Admin\AppData\Local\Temp/ipykernel_1708/4163829625.py:8: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  flight_used[column] = label_encoder.fit_transform(flight_used[column])
```

In [113]:
```
!pip install dmba
```

```
Requirement already satisfied: dmba in c:\programdata\anaconda3\lib\site-packag
es (0.1.0)

WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
```

In [139]:
```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
from dmba import plotDecisionTree, classificationSummary, regressionSummary
```

In [140]:
```
# split the train and validation data
x = flight_used.drop(['DepDelay', 'ArrDelay','DepDelayMinutes'], axis='columns')
y = flight_used['DepDelayMinutes']
train_x, val_x, train_y, val_y = train_test_split(x, y, test_size = .4, random_st
```

In [142]:
```
!pip install pydotplus
!conda install python-graphviz
!pip install graphviz
```

```
Requirement already satisfied: pydotplus in c:\programdata\anaconda3\lib\site-p
ackages (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in c:\programdata\anaconda3\lib
\site-packages (from pydotplus) (3.0.4)

WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)

^C
Requirement already satisfied: graphviz in c:\programdata\anaconda3\lib\site-pa
ckages (0.20.1)

WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn (c:\programdata\anaconda3\l
ib\site-packages)
```

In [143]:
```
tree = DecisionTreeClassifier(random_state=2,max_depth=1000, min_samples_split=6,
min_impurity_decrease=0.5,criterion="entropy")
tree.fit(train_x, train_y)
# get importance
importance = tree.feature_importances_
# summarize feature importance
impList = zip(train_x.columns, importance)
for feature in sorted(impList, key = lambda t: t[1], reverse=True):
    print(feature)
```

```
('Airline', 0.0)
('Origin', 0.0)
('Dest', 0.0)
('AirTime', 0.0)
('CRSElapsedTime', 0.0)
('ActualElapsedTime', 0.0)
('Distance', 0.0)
('Month', 0.0)
('DayofMonth', 0.0)
('TaxiOut', 0.0)
('WheelsOff', 0.0)
('WheelsOn', 0.0)
('TaxiIn', 0.0)
('DivAirportLandings', 0.0)
```

In [145]:
```
#Gradient Boost
from sklearn.ensemble import GradientBoostingRegressor

gbr1 = GradientBoostingRegressor(n_estimators = 100, random_state = 0)

gbr1.fit(train_x, train_y)

pred = gbr1.predict(val_x)
```

In [146]:
```
from sklearn.metrics import mean_squared_error
mse1=mean_squared_error(val_y,pred)
import math
rmse=math.sqrt(mse1)
rmse
```

Out[146]:
```
41.00278548128507
```

In [148]:
```
from sklearn.ensemble import RandomForestRegressor
# create regressor object
regressor1 = RandomForestRegressor(n_estimators = 100, random_state = 0)
regressor1.fit(train_x, train_y)
pred2=regressor1.predict(val_x)
```

In [149]:
```
from sklearn.metrics import mean_squared_error
mse=mean_squared_error(val_y,pred2)
import math
rmse=math.sqrt(mse)
rmse
```

Out[149]: 39.55441768185367