Building a Flight Delay Classification Model with Machine Learning

Harini Lakshmanan

Mohammad Mahmoudighaznavi

Department of Data Science, University of San Diego

ADS502: Data Mining

Dr. Tarshizi

August 12, 2022

Abstract

The airline industry plays an important and critical role in the world's transportation sector. Multiple businesses and industries too rely on the aviation industry to connect with various parts of the world. But varied factors not just including extreme weather conditions, air traffic control, airport operations, increased traffic volume, etc. have the capability to, directly and indirectly, affect airline services leading to delayed flights (McCarthy, 2022). Understanding the effect of the issue prior hand by predicting these delays to an extent, would allow the airline operators to be better prepared for the potential reasons for the delay of a flight in advance. If this data is also available to the consumers/passengers, it would also help them in planning their journey more efficiently.

The main objective of this project is to look at data mining approaches for delay prediction in the United States airline industry that occur due to a multitude of reasons. For this project, for the first part, we collected the dataset from Kaggle on Airline delay with eight different attributes of airlines related to delay and did a basic exploratory analysis and removed outliers to get clean data. For the next part, we mainly focused on applying data mining techniques like Random Forest classifier, Naive Bayes, Classification, and regression trees, C5.0 decision trees, etc, and compared the results between these models and derived the best algorithm for this problem.

Building a Flight Delay Classification Model with Machine Learning

In this project, we use historical on-time performance data to predict whether the arrival of a scheduled passenger flight will be delayed or not. We approach this problem as a classification problem, predicting two classes -- whether the flight will be delayed, or whether it will be on time. Broadly speaking, in machine learning and statistics, classification is the task of identifying the class or category to which a new observation belongs, on the basis of a training set of data containing observations with known categories. Since this is a binary classification task, there are only two classes. In this experiment, we train different models along with an outcome measure that indicates the appropriate category or class for each model. We are using R as our tool for all steps in building these models. Our first step in this process was to decide how to focus the scope of our analysis.

Data description

The Airline delay data set was downloaded from the Kaggle dataset, Airline dataset to predict delay (Chacko, 2022) and had 539,383 instances and nine different variable features namely ID, Airline, Flight, AirportFrom, AirportTo, DayOfWeek, Time, Length, and Delay. Since ID does not have any relationship with a flight delay, the remaining seven variables were taken into consideration initially for the delay prediction. Brief list of the data set is shown in Table1. We quickly realized that the 600 thousand flights we had data for would make quick testing very difficult and our computers are not able to process the models for this size of data set. So, in order to speed up our testing, we decided to break up our data into busiest airports that have over 5000 flights from those airports which reduce the size of our data set to almost 140 thousand flights.

Table 1

Brief list of the data set

	id	Airline	Flight	AirportFrom	AirportTo	DayOfWeek	Time	Length	Delay
1	1	CO	269	SFO	IAH	Wednesday	15	205	Delayed
2	2	US	1558	PHX	CLT	Wednesday	15	222	Delayed
3	3	AA	2400	LAX	DFW	Wednesday	20	165	Delayed
4	4	AA	2466	SFO	DFW	Wednesday	20	195	Delayed
5	5	AS	108	ANC	SEA	Wednesday	30	202	OnTime
6	6	CO	1094	LAX	IAH	Wednesday	30	181	Delayed

First Stage Data Exploration of Airline Data

It is important for data scientists to perform exploratory data analysis to analyze the data before coming into any form of assumptions. It helps in comprehending the structure of the dataset and helps in making the data modeling process easier. There were 18 unique airlines flying across the United States used for this delay classification analysis. Based on the summary statistics in Table 2 for the dataset, the average flight time was around 802 minutes and the average flight Length was around 132.

Table 2

Summary statistics for the Airline Delay Dataset

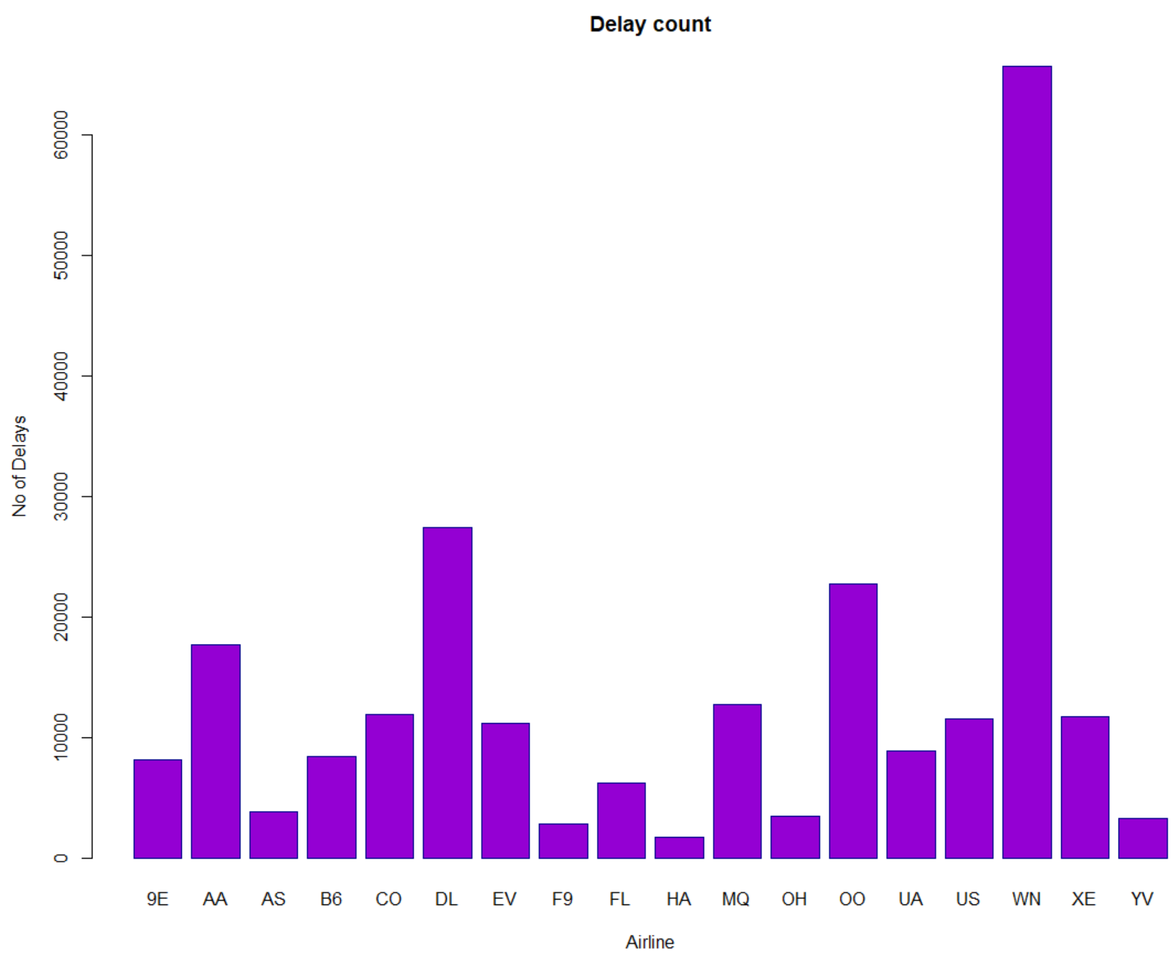
Airline	Flight	AirportFrom	AirportTo	DayOfWeek	Time	Length
Length:539383	Min. : 1	Length:539383	Length:539383	Length:539383	Min. : 10.0	Min. : 0.0
Class :character	1st Qu.: 712	Class :character	Class :character	Class :character	1st Qu.: 565.0	1st Qu.: 81.0
Mode :character	Median :1809	Mode :character	Mode :character	Mode :character	Median : 795.0	Median :115.0
	Mean :2428				Mean : 802.7	Mean :132.2
	3rd Qu.:3745				3rd Qu.:1035.0	3rd Qu.:162.0
	Max. :7814				Max. :1439.0	Max. :655.0

Arrival delay time and departure delay time would have given a more descriptive understanding of the delay, but this data was not available as a part of this dataset as the distribution of the delay time would have helped us in understanding whether the flights arrive before the published schedule and leave before or after the published schedule and would have given more depth to understanding and prediction of delay.

As mentioned before, there are in total 18 airlines operating in the United States used for this data set and Southwest has the greatest number of flights delayed and can be seen in Figure 1.

Figure 1

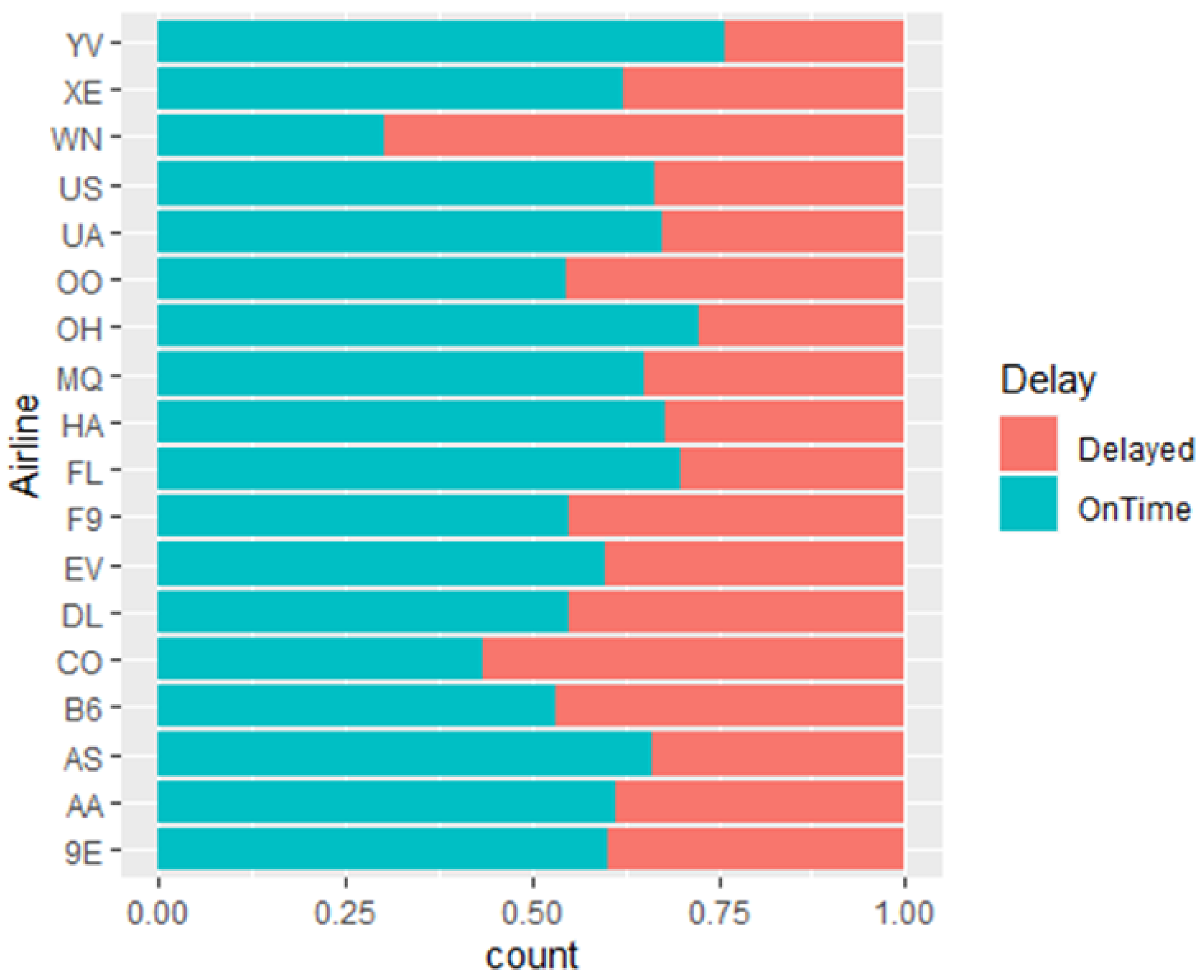
Airline vs Delay count



By normalizing the airline with delay overlay we can understand the distribution of the percentage of each airline that had delayed. Figure 2 shows the comparison of this normalization in a bar graph. As we see in this figure, the WN airline has the most portion delays among other airlines as well. On the other hand, YV has the least proportion of delayed flights.

Figure 2

Normalized bar graph of Airline with Delay overlay



Comparing the number of flights delayed based on the departing and arriving airports, from Figure 3 and Figure 4, ATL (Hartsfield-Jackson Atlanta) and ORD (O'Hare International Airport Chicago) have the most delays for both arriving and departing airports. Extended delays mainly occur in these airports due to the sheer traffic volume of flights departing from and arriving at these two airports.

Figure 3

AirportFrom vs Delay

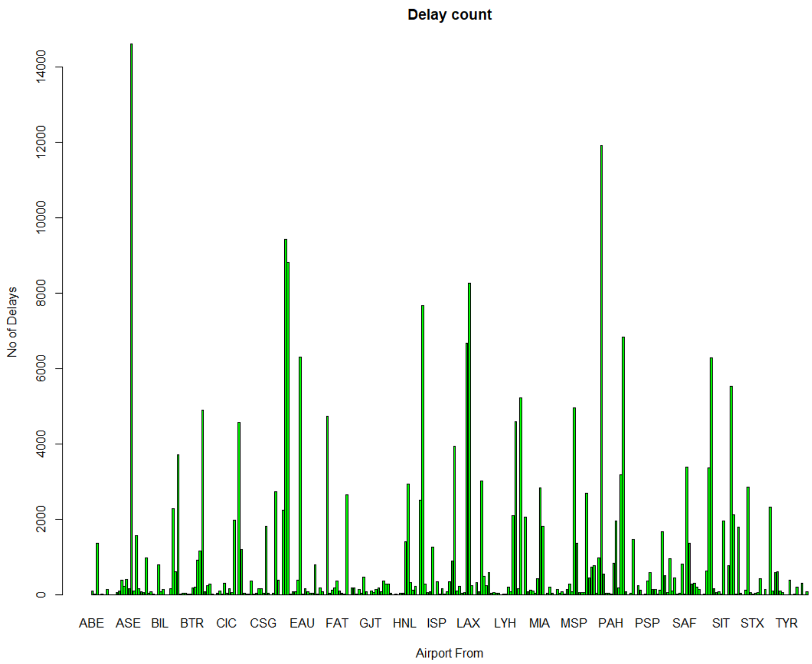
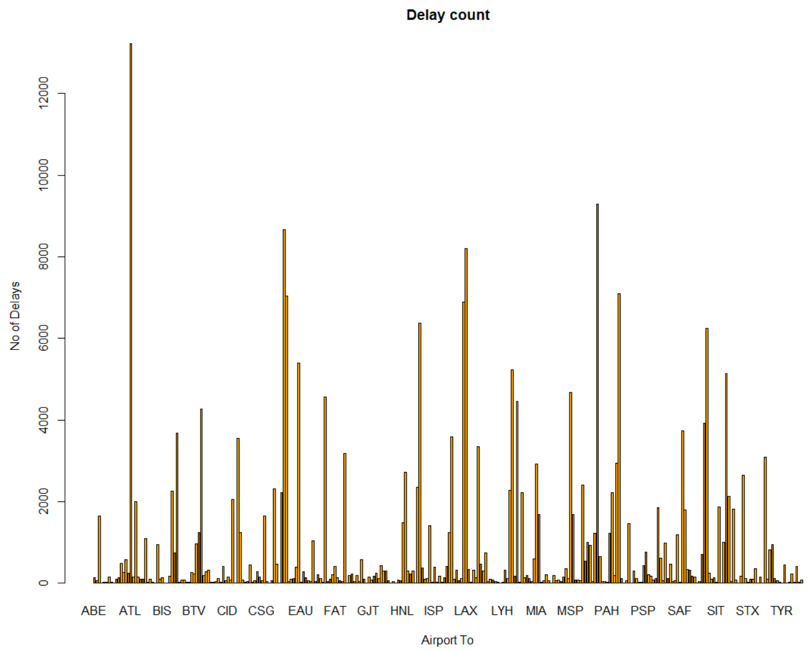


Figure 4

AirportTo vs Delay



The normalized bar graphs of AirportFrom and AirportTo with Delay overlay are shown in figure 5 and figure 6 to compare the percentage of Delay in each airport. The results show that MDW airport has the greatest proportion delayed flights and DCA has the least delay proportion.

Figure 5

Normalized bar graph of AirportFrom with Delay overlay

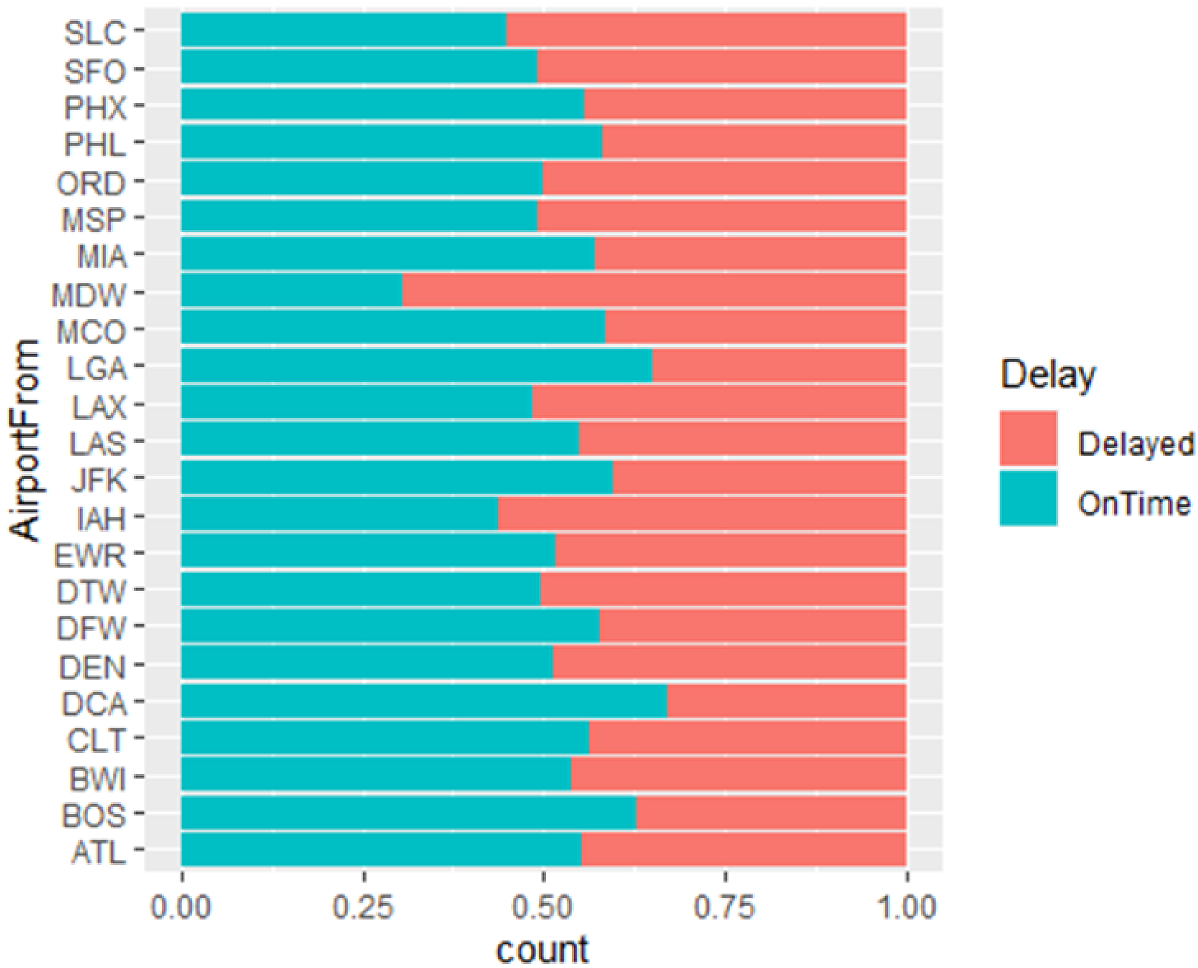
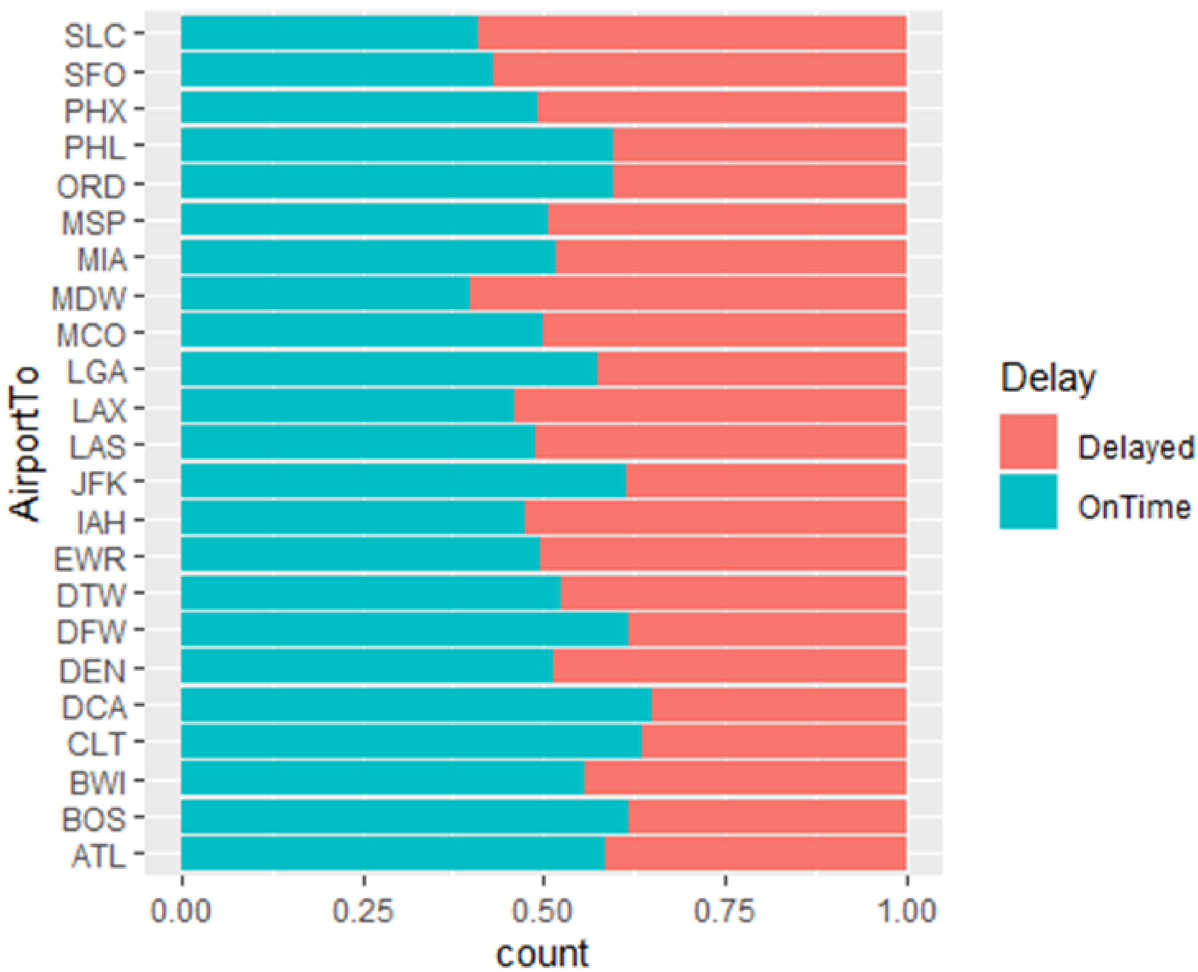


Figure 6

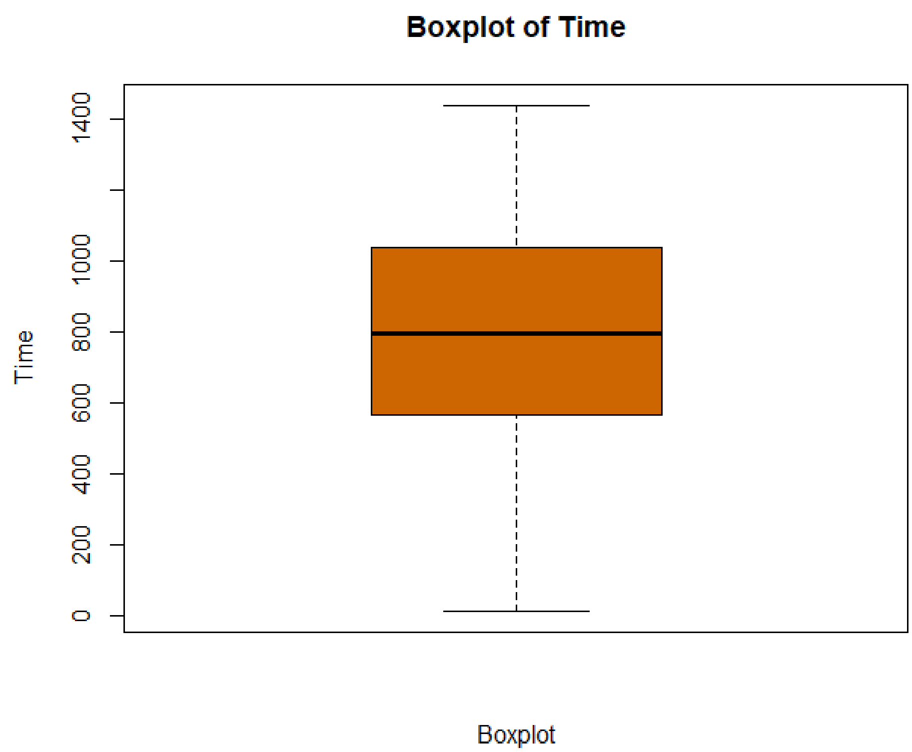
Normalized bar graph of AirportTo with Delay overlay



There are two main nominal variables in this study namely Time and Length of flight. To understand the outliers, we need to plot a box plot of these variables, which helps us realize the distribution of these variables which would be helpful in outlier detection as a first step toward the data cleaning process. Based on the box plot of Time, from Figure 7, there are no outliers as the variables are well within 1.5 times the interquartile range, which is generally assumed for outliers.

Figure 7

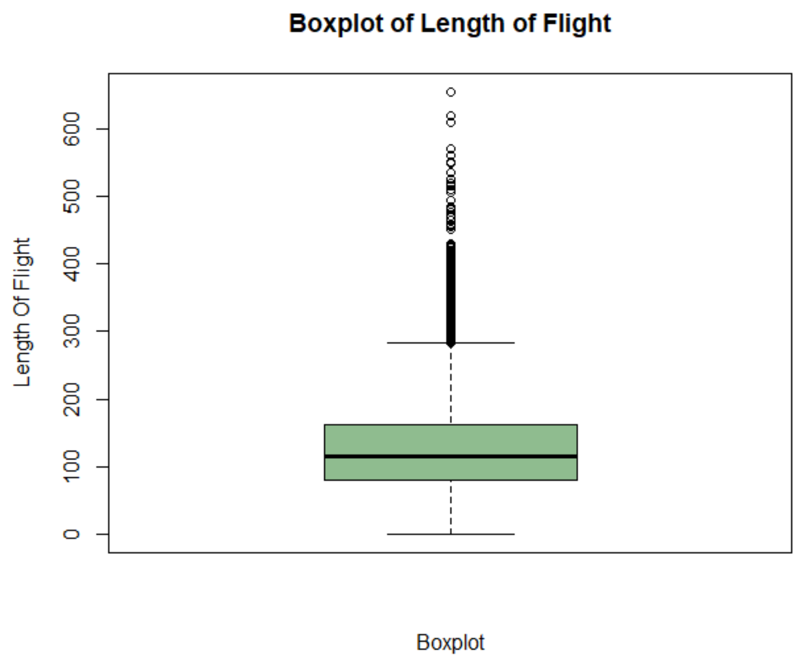
Box plot of Time variable



For length variables, based on Figure 8, there are a huge number of outliers. The range of the Length of flight is 655 and 1.5 times Q3 is around 300. Hence we must plan for addressing the outliers in the preprocessing stage.

Figure 8

Box plot of Length of Flight



Data preprocessing

Data preprocessing is an important part of a data mining project as it helps in eliminating the inconsistencies in the data, resolving missing data, and dealing with outliers, which can negatively affect the accuracy of a model. There are four major steps in data preparation or preprocessing.

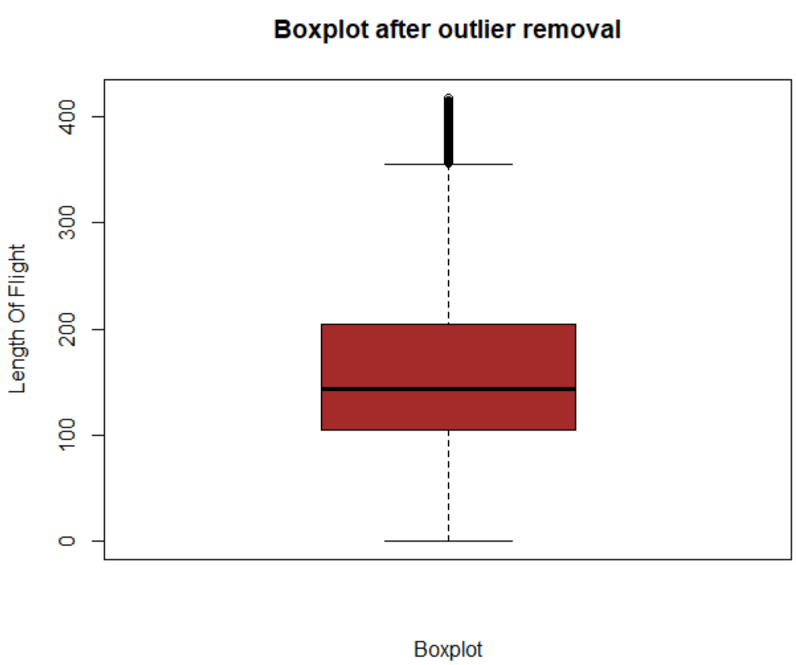
1. Data Quality Assessment
2. Data Cleaning
3. Data Transformation
4. Data reduction

Since our data was already well-refined from Kaggle we did not have any missing data or noisy data in the dataset and hence we did not have the need to do an extensive data cleaning process. Most of our effort in the process was mainly concentrated on Data Quality Assessment and mainly understanding the outliers and removing them as it might lead to an inaccurate model.

From our exploratory data analysis, we found out that the Length of the flight has significant outliers that needs to be addressed. After understanding the airports from which the outliers are present, the airports were filtered out in R by narrowing down the dataset based on over five thousand flights from the departure airports. The reduced data sent has 123,464 records from after the outlier removal. Plotting the box plot of Length of flight again after outlier removal in Figure 9, you can see that the outliers have been significantly reduced compared to

Figure 9

Boxplot of length of flight after outlier removal



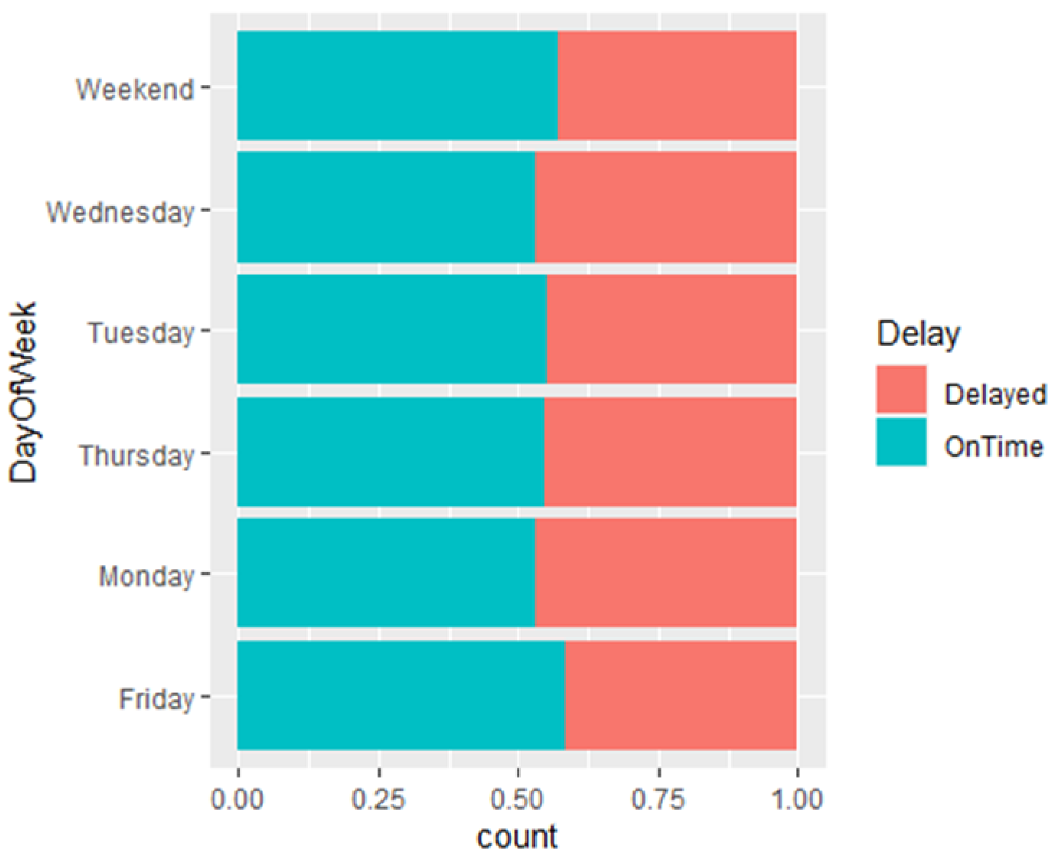
For the initial study, we did not perform any data transformation like normalization for this project as the other variables were mostly categorical variables and initially we did not use any algorithm that does not make assumptions about the distribution of your data, such as k-nearest neighbors and artificial neural networks. Normalization was done separately for Neural Networks when we performed the algorithm for the nominal variables.

Second Stage Data Exploration of Airline Data

After our preprocessing stage with the clean dataset, as a first exploratory examination, we watched the likelihood of delay for the whole dataset based on the Days of the week. The best approach is to plot a normalized bar chart of delay independently which can be seen in Figure 10. It can be observed that the probability of delays was lower on the weekends and closer to the weekends, specifically Friday compared to the mid-weekdays.

Figure 10

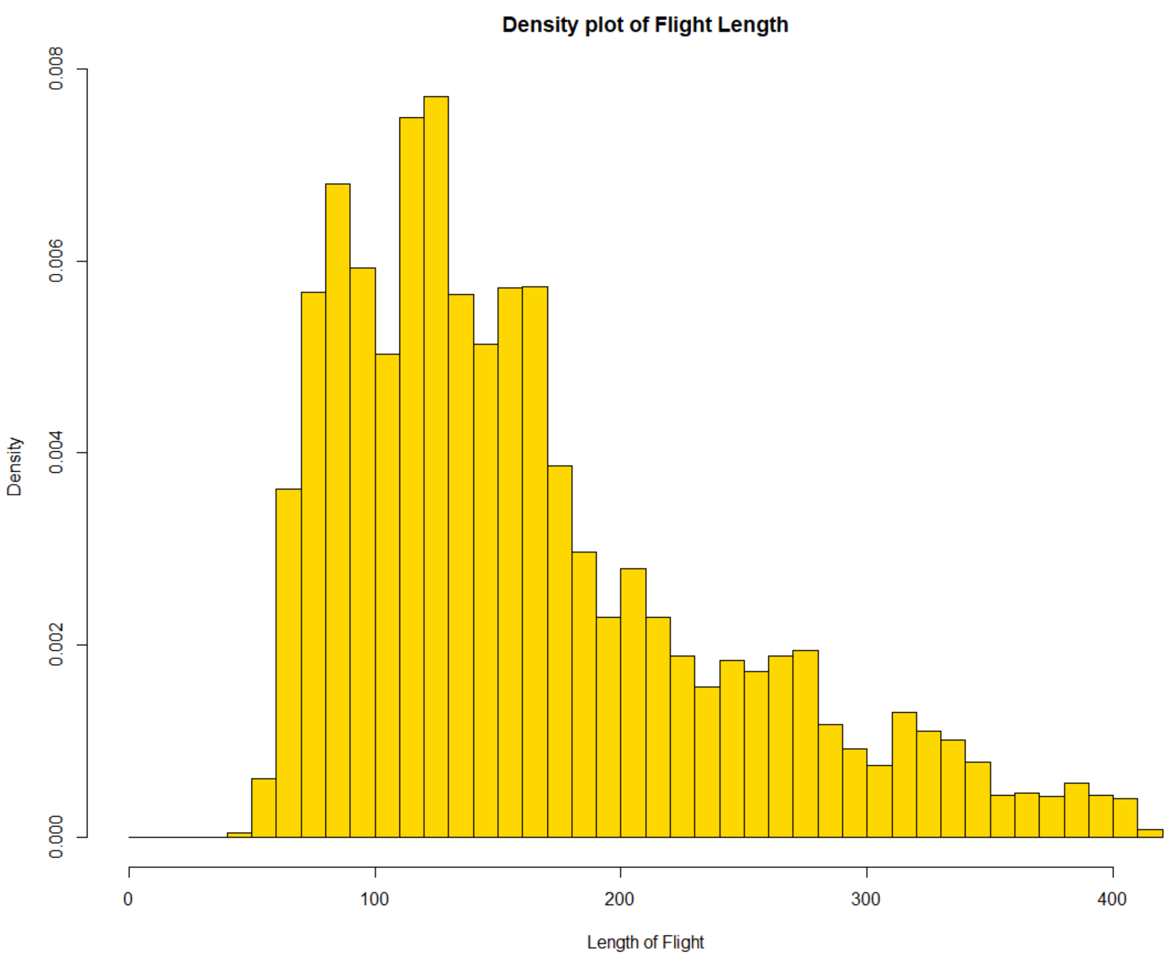
Normalized bar graph of DayOfWeek with Delay overlay



To understand the relationship between Length of flight and Delay, The probability of the length of flight over the dataset would help in understanding the concentration of the length of the flight. From Figure 11 it can be understood that the most common length of flight is between 80 and 180 and it is expected to have the most delays as it is more frequent.

Figure 11

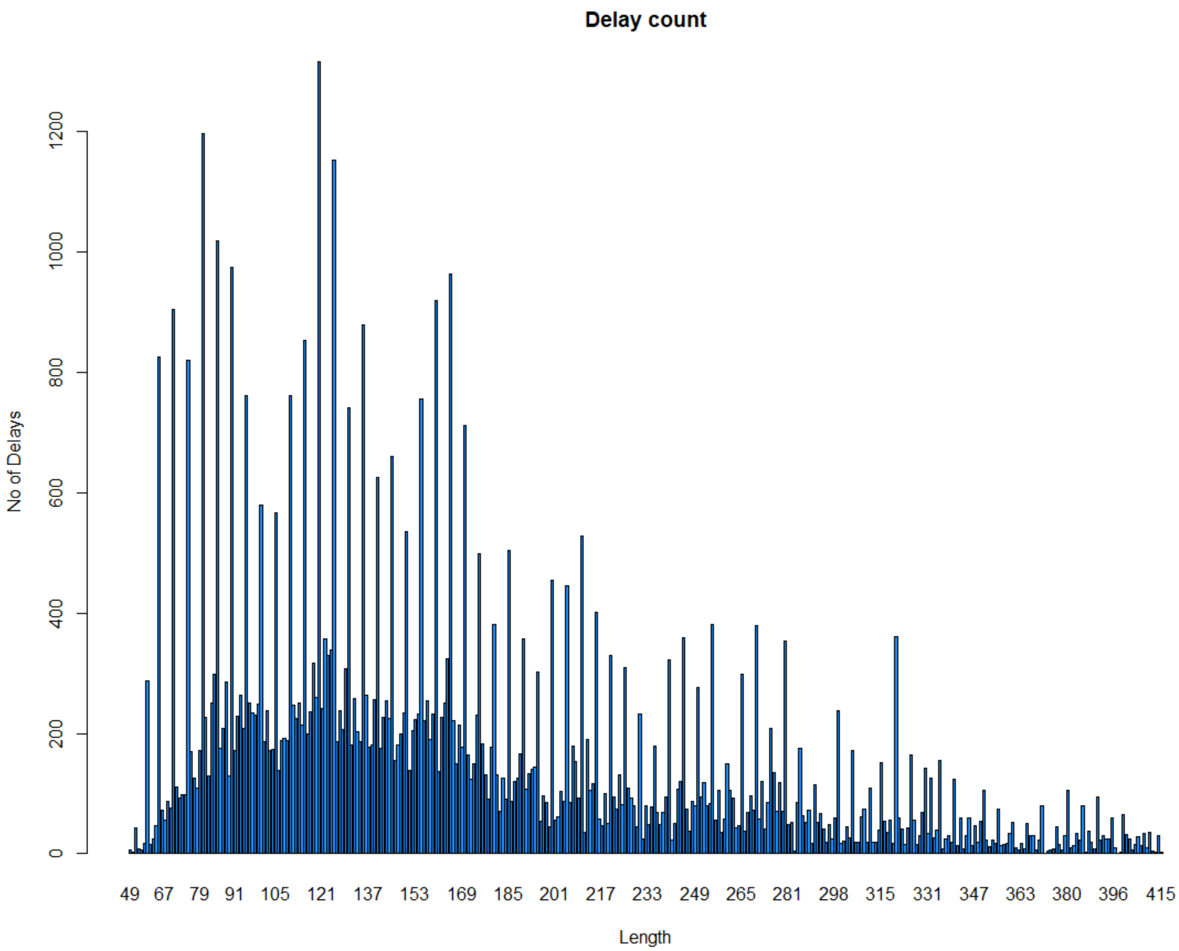
Probability density plot of Length of Flight



From Figure 12, the length of flight is plotted against Delay count to better understand the relationship between these two variables, and the above statement of the length of flight between 80 and 180 having the greatest number of delays can be visualized and confirmed.

Figure 12

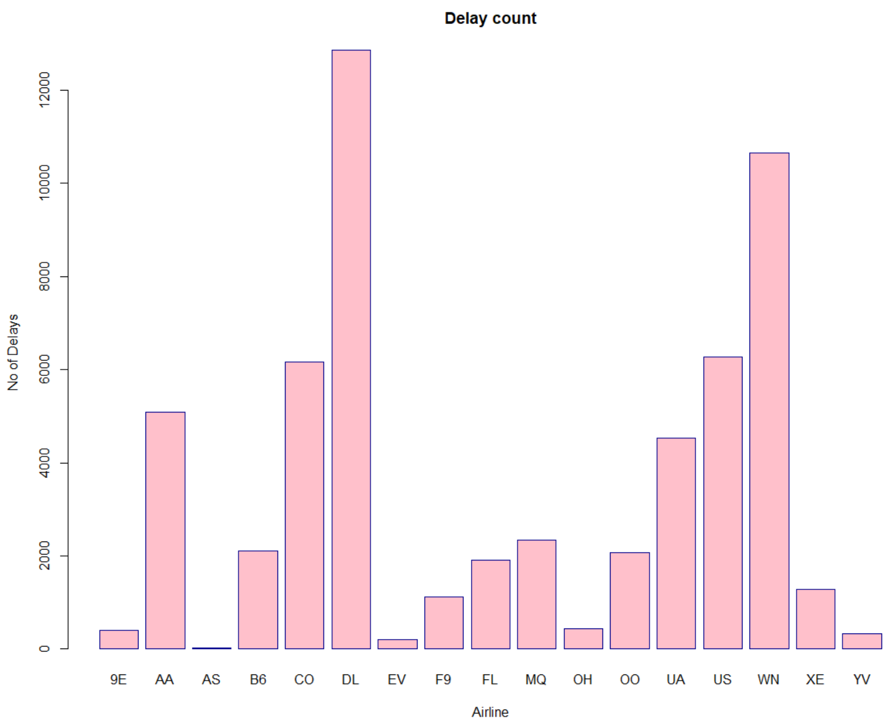
Delay count vs Length of Flight



Understanding the relationship between the airline and the number of delays, Delta and Southwest have the greatest number of delays from Figure 13. Delta has a huge number of delays only due to the sheer number of flights they operate in the United States. To better understand the relationship, the normalized bar graph between these two variables is plotted in Figure 2 previously, and still Southwest had the most delays per number of flights operated followed by Continental Airlines.

Figure 13

Delay count vs Airline

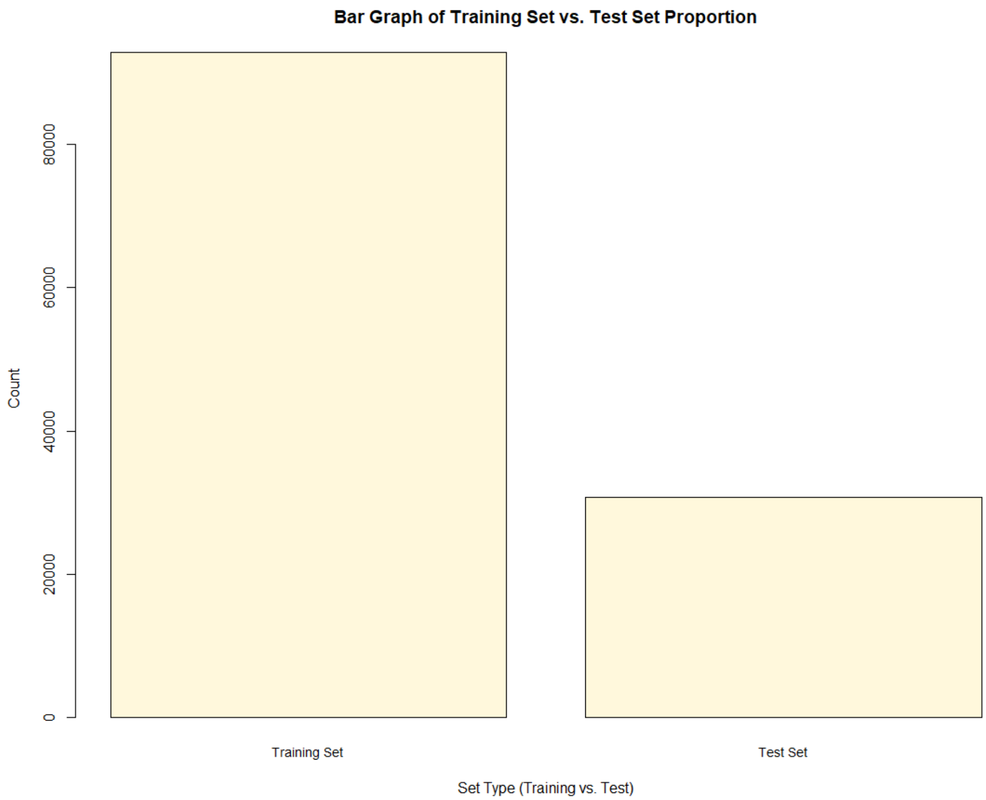


Test and Train Split

In order to evaluate and estimate the performance of the data mining algorithm, we performed the train-test split procedure in R as we have a big data set and would help in understanding the model performance by evaluating the training method on the test dataset pretty quickly. We chose a 75%-25% split for train and test data respectively and the bar chart of the frequency count of the split can be found in Figure 14. It is important to perform this to validate the performance of the model on the new data. The data set was split into 92,770 records for the training set and 30,694 for the test set.

Figure 14

Bar Chart of Frequency count between Train and Test data



Modeling

We are going to run the models through different algorithms. These models will run through the training data set and then will be evaluated by testing the data set. In each algorithm, we use different performance metrics to evaluate our model. These performance metrics will be calculated based on the formula provided in Table 3 and will be used to compare different methods.

Table 3

Performance metrics and their formula

Performance Metric	Formula
Accuracy	$Accuracy = \frac{TN + TP}{TN + TP + FN + FP}$
Precision	$Precision = \frac{TP}{TP + FP}$
Recall	$Recall = \frac{TP}{TP + FN}$
F1 Score	$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$

Decision Tree

Decision Tree is a machine-learning algorithm that can be a classification or regression tree analysis. The decision tree can be represented by graphical representation as a tree with leaves and branch's structure. The leaves are generally the data points and branches are the condition to make decisions for the class of data set.

a. CART Algorithm

The CART method produces decision trees that are strictly binary, containing exactly two branches for each decision node. CART recursively partitions the records in the training data set into subsets of records with similar values for the target attribute. The CART algorithm grows the tree by conducting for each decision node an exhaustive search for all available variables and all possible splitting values, selecting the optimal split according to the Gini Index (from Kennedy et al.). We used the most effective attributes in our CART model including Airline, Time, AirportFrom, and DayOfWeek, and the results are shown in Table 4.

Table 4

Performance metrics for CART algorithm method

Performance Metrics (CART)	Value
Accuracy	0.624
Precision	0.504
Recall	0.618
F1 Score	0.555

b. C5.0 Algorithm

The C5 algorithm is J. Ross Quinlan’s extension of his own C4.5 algorithm for generating decision trees. Unlike CART, the C5.0 algorithm is not restricted to binary splits. This algorithm uses the concept of information gain or entropy reduction to select the optimal split. We used the same attributes to build a C5.0 model for this project. Table 5 shows the result of this method.

Table 5

Performance metrics for C5.0 algorithm method

Performance Metrics (C5.0)	Value
Accuracy	0.637
Precision	0.510
Recall	0.639
F1 Score	0.567

As we can see from the results in Table 4 and Table 5, the C5.0 method has slightly better results but it is not good enough to count on this model.

Random Forests

Random forests build a series of decision trees and combine the trees and disparate classifications of each record into one final classification. Random forests are an example of an ensemble method. The random forests algorithm begins building each decision tree by taking a random sample, with replacement, from the original training data set. In this way, each tree will have a different data set on which to be built. For each node of the decision tree, a subset of predictor variables is selected for consideration. Once the different trees are built, they are used to classify the records in the original training data set (Liu et al., 2012). Every record in the data set is given a classification by every tree. Since these classifications are highly unlikely to be unanimous for all records, each classification is considered a vote for that particular target variable value. The value with the largest number of votes is deemed the final classification of the record. Random forests results are shown in Table 6 below.

Table 6

Performance metrics for Random Forests algorithm method

Performance Metrics (RF)	Value
Accuracy	0.828
Precision	0.806
Recall	0.876
F1 Score	0.840

Table 6 shows that the random forests algorithm works much better in our case.

Naive Bayes Classifier

Naive Bayes classification is a probabilistic classification method based on the application of the Bayes theorem. It computes the a-posterior probabilities of the categorical class variable using the Bayes rule from the given independent predictor variables (Tan et al., 2019). It also computes the probability of numerical variables of each class using the mean and the standard deviation of each variable. One of the main assumptions in Naive Bayes is that it assumes that all the variables are unrelated to each other which is rare in real life and hence we must make sure that the variables used for the algorithm are independent to get the best out of the classifier. Also if there is a category in the test data set that was not present in the training set it assigns zero probability to it. Hence, we must make sure that all the categories are present in both the training and test data set before applying the Naive Bayes method. We used the most effective attributes in our Naive Bayes model which are Airline, Time, AirportFrom, and DayOfWeek and the results are shown in Table 7.

Table 7

Performance metrics for Naive Bayes Classifier method

Performance Metrics (RF)	Value
Accuracy	0.624
Precision	0.501
Recall	0.619
F1 Score	0.554

Neural Network Classifier

Neural network classifier imitates non-linear learning similar to neurons present in nature. The main advantage of this approach for classification is its robustness to noise in the data as well as non-linearity (Knocklein, 2019). One of the aspects of neural networks is that it works better if the numerical data are normalized. Hence, we normalized the Time and Length data before proceeding to perform the Neural Network analysis. Table 8 presents the results of the model by using the effective attributes the same as Naive Bayes and it has only similar results to the Naive Bayes classifier.

Table 8

Performance metrics for Neural Network Classifier

Performance Metrics (RF)	Value
Accuracy	0.638
Precision	0.537
Recall	0.632
F1 Score	0.581

Conclusion

In this paper we performed different classification models for flight delays by adapting it into machine learning problems. Five algorithms were used for delay classification and all of them were used for algorithm performance evaluation. After applying classifiers to delay classification, the values of the measures were compared to evaluate the performance of each model.

The results show that the highest value of accuracy, precision, recall, and F1 score are generated by Random Forests model. Such high values indicate that the Random Forests performs well to classify our data set to either Delayed or Ontime classes.

References

Chacko, J. (2022, June 19). *Airlines dataset to predict a delay*. Kaggle. Retrieved August 13, 2022, from https://www.kaggle.com/datasets/jimschacko/airlines-dataset-to-predict-a-delay

Knocklein, O. (2019, June 15). *Classification using neural networks.* Medium. Retrieved August 11, 2022, from https://towardsdatascience.com/classification-using-neural-networks-b8e98f3a904f#:~:text=Summary,passing%20outputs%20to%20further%20layers.

Larose, C. D., & Larose, D. T. (2019). *Data Science Using Python and R.* Wiley.

Liu, Y., Wang, Y., & Zhang, J. (2012, September). New machine learning algorithm: Random forest. In *International Conference on Information Computing and Applications* (pp. 246-252). Springer, Berlin, Heidelberg.

McCarthy, D. (2022, April 8). *What are the most common reasons for flight delays in the U.S.?* Travel Market Report: The Voice of The Travel Advisor. Retrieved August 10, 2022, from https://www.travelmarketreport.com/articles/What-Are-the-Most-Common-Reasons-For-Flight-Delays-in-the-US

Tan, P.-N., Steinbach, M., Karpatne, A., & Kumar, V. (2019). *Introduction to Data Mining(2nd ed.)*. Pearson.

Code Appendix

```
#Load airline dataset
df <- read.csv(file.choose(), header= T)
#replacing variables of Dayofweek
df['DayOfWeek'][df['DayOfWeek']==1] <- "Monday"
df['DayOfWeek'][df['DayOfWeek']==2] <- "Tuesday"
df['DayOfWeek'][df['DayOfWeek']==3] <- "Wednesday"
df['DayOfWeek'][df['DayOfWeek']==4] <- "Thursday"
df['DayOfWeek'][df['DayOfWeek']==5] <- "Friday"
df['DayOfWeek'][df['DayOfWeek']==6] <- "Weekend"
df['DayOfWeek'][df['DayOfWeek']==7] <- "Weekend"

head(df)

##   id Airline Flight AirportFrom AirportTo DayOfWeek Time Length Del
ay
## 1  1      CO    269         SFO       IAH Wednesday   15    205
1
## 2  2      US   1558         PHX       CLT Wednesday   15    222
1
## 3  3      AA   2400         LAX       DFW Wednesday   20    165
1
## 4  4      AA   2466         SFO       DFW Wednesday   20    195
1
## 5  5      AS    108         ANC       SEA Wednesday   30    202
0
## 6  6      CO   1094         LAX       IAH Wednesday   30    181
1

dim(df)

## [1] 539383      9

library(dplyr)

## 
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
## 
##     filter, lag

## The following objects are masked from 'package:base':
## 
##     intersect, setdiff, setequal, union

glimpse(df)
```

```
## Rows: 539,383
## Columns: 9
## $ id          <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16, 17,~
## $ Airline     <chr> "CO", "US", "AA", "AA", "AS", "CO", "DL", "DL",
"DL", "AA"~
## $ Flight      <int> 269, 1558, 2400, 2466, 108, 1094, 1768, 2722, 2
606, 2538, ~
## $ AirportFrom <chr> "SFO", "PHX", "LAX", "SFO", "ANC", "LAX", "LAX"
, "PHX", "S~
## $ AirportTo   <chr> "IAH", "CLT", "DFW", "DFW", "SEA", "IAH", "MSP"
, "DTW", "M~
## $ DayOfWeek   <chr> "Wednesday", "Wednesday", "Wednesday", "Wednesd
ay", "Wedne~
## $ Time        <int> 15, 15, 20, 20, 30, 30, 30, 30, 35, 40, 49, 50,
50, 55, 55~
## $ Length      <int> 205, 222, 165, 195, 202, 181, 220, 228, 216, 20
0, 201, 212~
## $ Delay       <int> 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0,
0, 0, 0, 0~
```

```
summary(df)
```

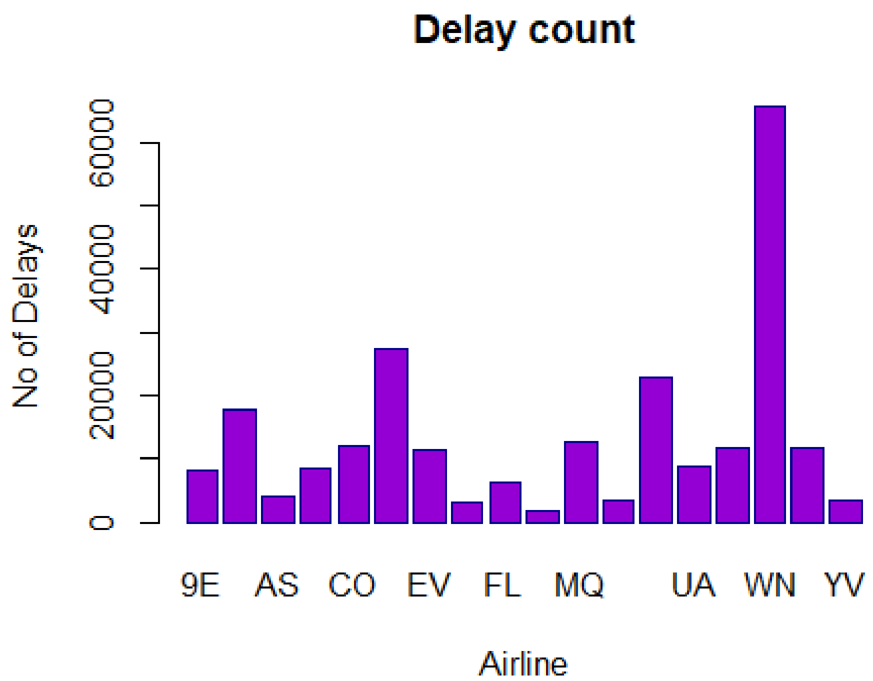
```
##        id           Airline               Flight     AirportFrom
##  Min.   :     1   Length:539383      Min.   :   1   Length:539383
##  1st Qu.:134847   Class :character   1st Qu.: 712   Class :characte
r
##  Median :269692   Mode  :character   Median :1809   Mode  :characte
r
##  Mean   :269692                      Mean   :2428
##  3rd Qu.:404538                      3rd Qu.:3745
##  Max.   :539383                      Max.   :7814
##   AirportTo          DayOfWeek              Time            Length
##  Length:539383      Length:539383      Min.   :  10.0   Min.   :  0
.0
##  Class :character   Class :character   1st Qu.: 565.0   1st Qu.: 81
.0
##  Mode  :character   Mode  :character   Median : 795.0   Median :115
.0
##                                        Mean   : 802.7   Mean   :132
.2
##                                        3rd Qu.:1035.0   3rd Qu.:162
.0
##                                        Max.   :1439.0   Max.   :655
.0
##      Delay
```

```
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.4454
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

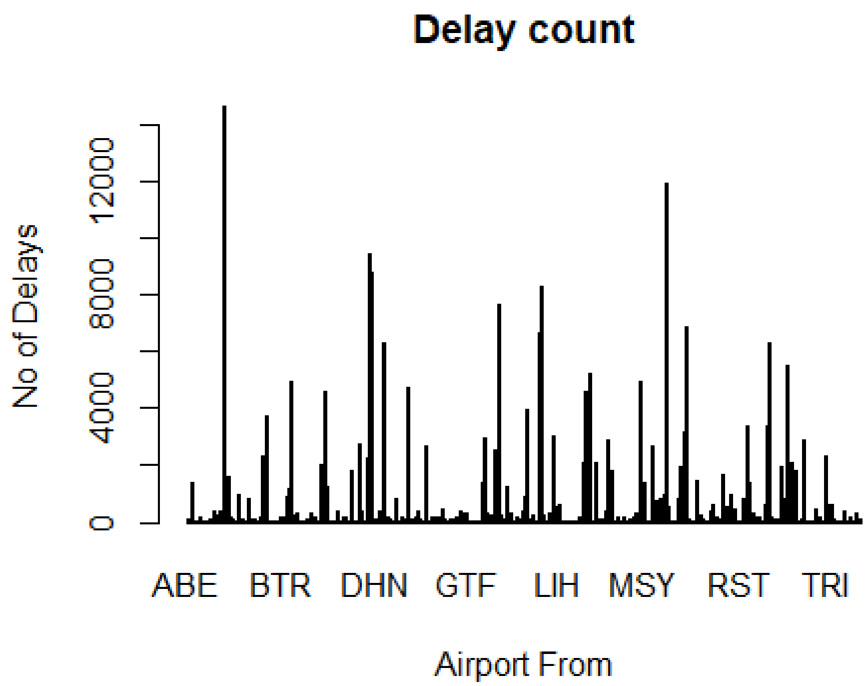
```
#total number of ailines
airlineNo <- unique(df$Airline)
glimpse(airlineNo)
```

```
##  chr [1:18] "CO" "US" "AA" "AS" "DL" "B6" "HA" "OO" "9E" "OH" "EV"
"XE" ...
```

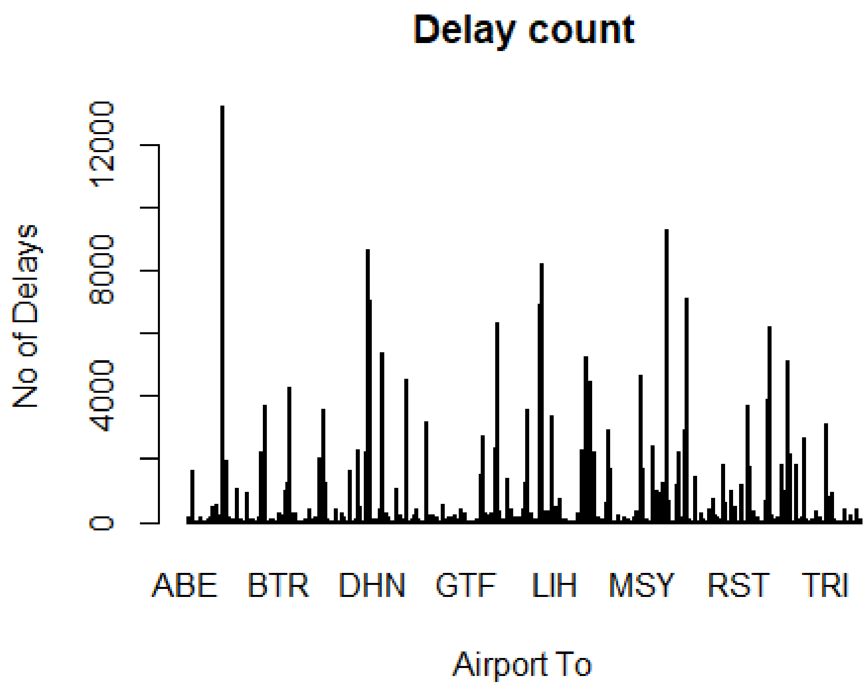
```
#Exploratory Data Analysis part 1

#Checking airlines with most and least delay
library(dplyr)
df_dt <- df[df$Delay==1,]
df_t <- df_dt %>% count(df_dt$Airline ,df_dt$Delay)
barplot(height = df_t$n,
        main = "Delay count",
        xlab = "Airline",
        ylab = "No of Delays",
        names.arg = df_t$`df_dt$Airline`,
        border = "dark blue", col = "darkviolet")
```



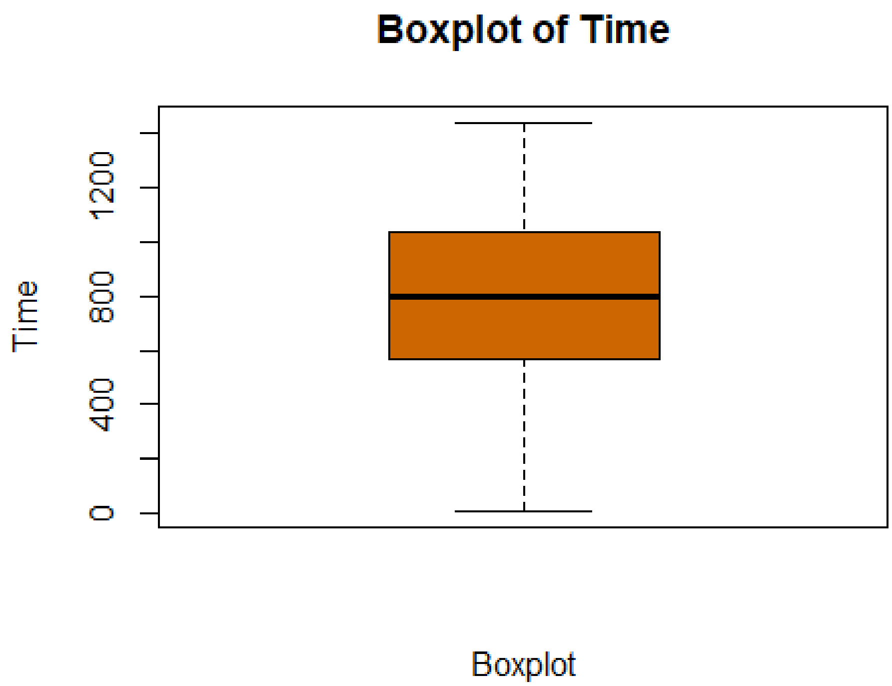

```
df_t[which.max(df_t$n),][1,1]

## [1] "WN"

sprintf("The airline with the most delay is Southwest: %s", df_t[which
.max(df_t$n),][1,1])

## [1] "The airline with the most delay is Southwest: WN"

#Checking Airport From with most and least delay
df_af <- df_dt %>% count(df_dt$AirportFrom ,df_dt$Delay)
barplot(height = df_af$n,
        main = "Delay count",
        xlab = "Airport From",
        ylab = "No of Delays",
        names.arg = df_af$`df_dt$AirportFrom`,  col = "green")
```



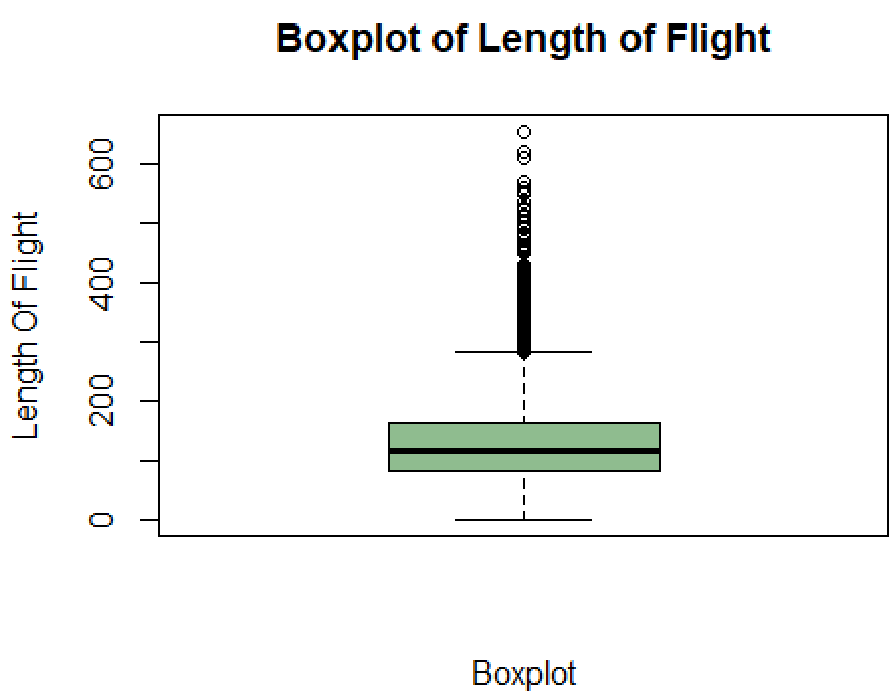

```
df_af[which.max(df_af$n),][1,1]

## [1] "ATL"

df_af <- df_af[order(-df_af$n),]
sprintf("%s (Hartsfield-Jakson Atalanta) and %s (O'Hare International
Airport Chicago) has the most delays for arriving airport", df_af[1,1]
, df_af[2,1])

## [1] "ATL (Hartsfield-Jakson Atalanta) and ORD (O'Hare International
Airport Chicago) has the most delays for arriving airport"

#Checking Airport From with most and least delay
df_at <- df_dt %>% count(df_dt$AirportTo ,df_dt$Delay)
barplot(height = df_at$n,
        main = "Delay count",
        xlab = "Airport To",
        ylab = "No of Delays",
        names.arg = df_at$`df_dt$AirportTo`,  col = "orange")
```



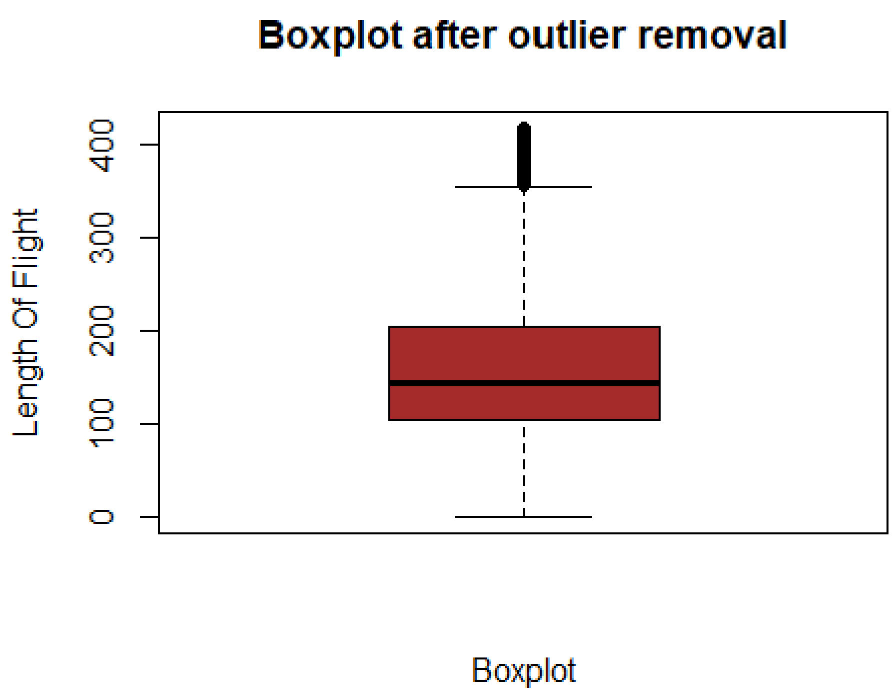

```
df_at[which.max(df_at$n),][1,1]

## [1] "ATL"

df_at <- df_at[order(-df_at$n),]
sprintf("%s (Hartsfield-Jakson Atalanta) and %s (O'Hare International
Airport Chicago) has the most delays for arriving airport", df_at[1,1]
, df_at[2,1])

## [1] "ATL (Hartsfield-Jakson Atalanta) and ORD (O'Hare International
Airport Chicago) has the most delays for arriving airport"

#boxplot of Time
boxplot(df$Time, xlab="Boxplot", ylab="Time", main="Boxplot of Time",
col = "darkorange3")
```



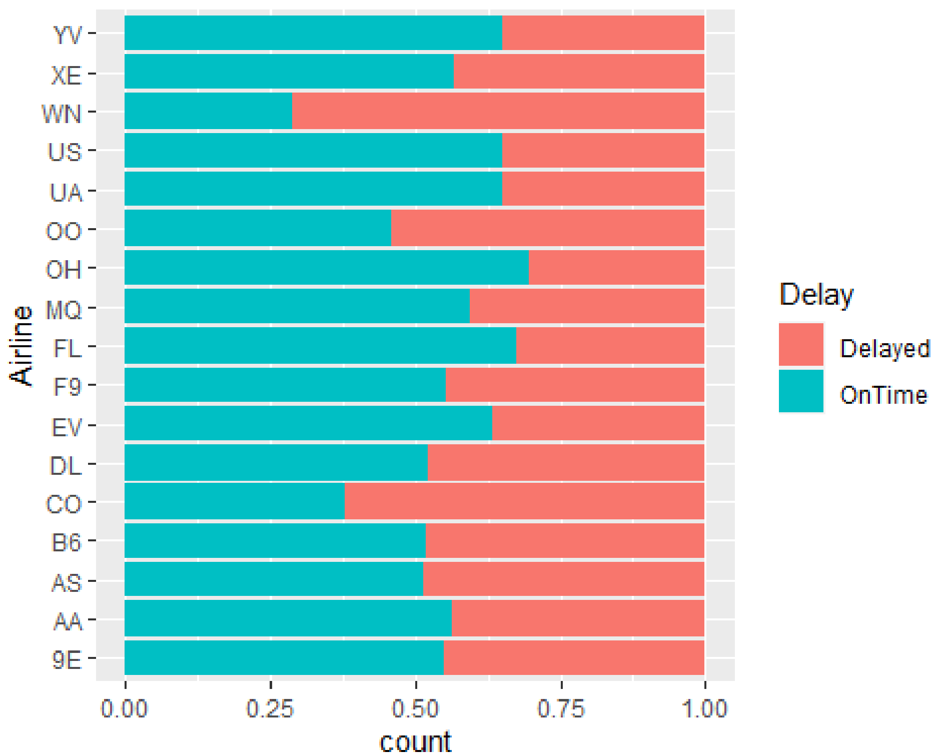

```r
#boxplot of length
boxplot(df$Length, xlab="Boxplot", ylab="Length Of Flight", main="Boxp
lot of Length of Flight", col = "darkseagreen")
```



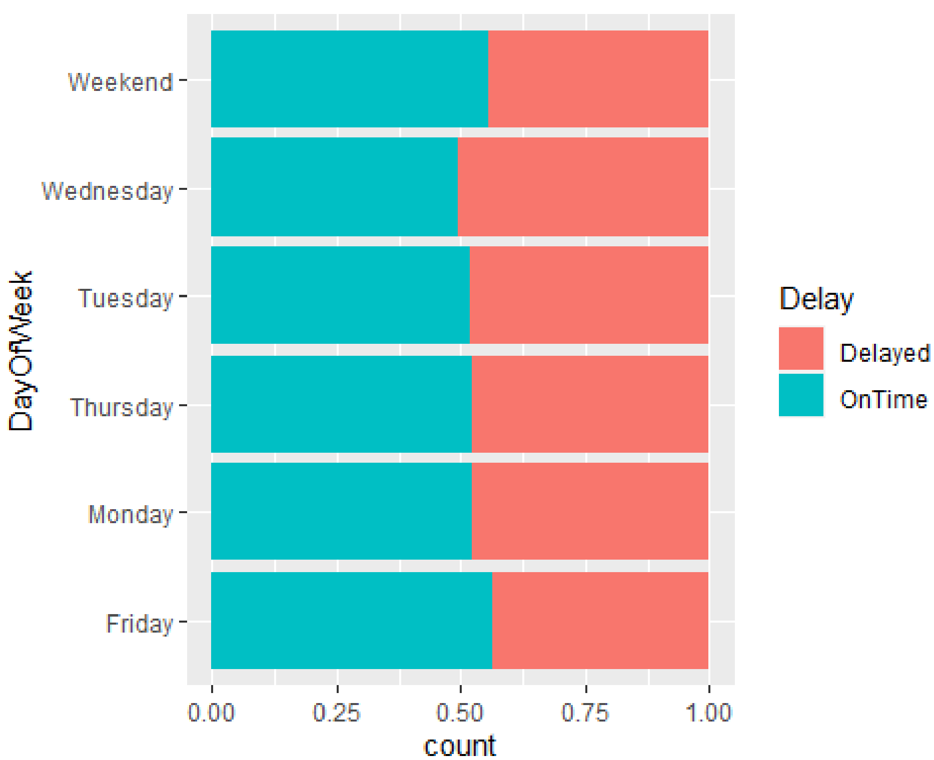
```
#Narrow down dataset based on over 5000 flight from departure airport
airports<-c("ATL","ORD","dfW","DEN","LAX","IAH","PHX","DTW","MCO","SLC
","MSP","EWR","ORD","JFK","SFO","CLT","LAS","BOS","MIA","MDW","DCA","P
HL","BWI","LGA")
df<-subset(df,AirportFrom %in% airports & AirportTo %in% airports )

dim(df)

## [1] 123464      9

#replacing variables of Delay
df['Delay'][df['Delay']==0] <- "OnTime"
df['Delay'][df['Delay']==1] <- "Delayed"


#boxplot of length
boxplot(df$Length, xlab="Boxplot", ylab="Length Of Flight", main="Boxp
lot after outlier removal", col = "brown")

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.1.3
```



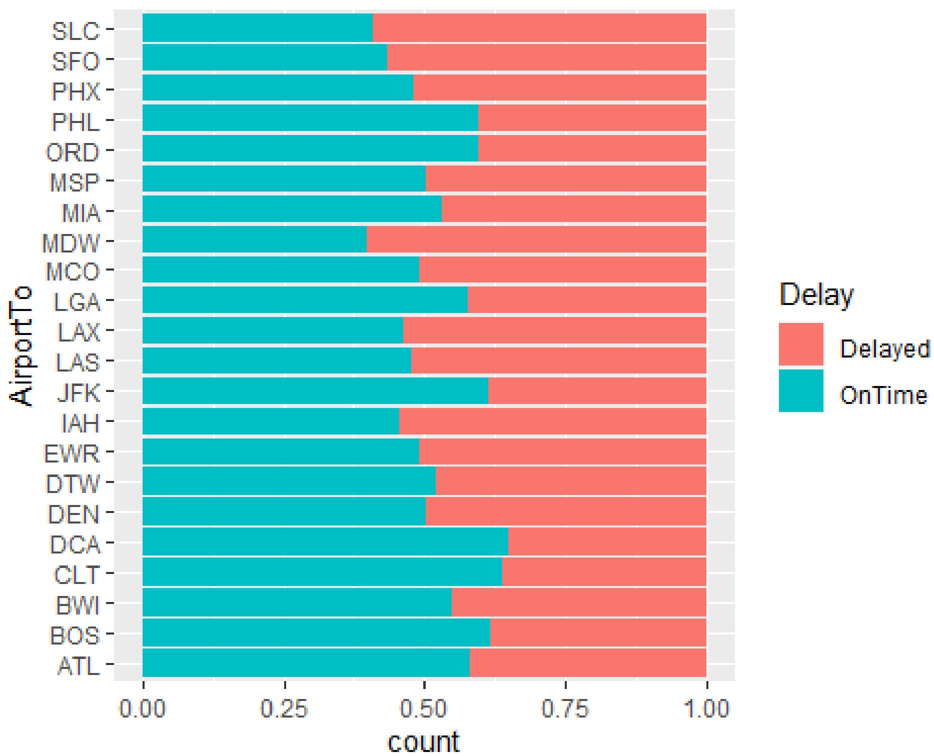

```
#normalized bar graph of airline with delay overlay
ggplot(df,aes(Airline))+geom_bar(aes(fill=Delay),position="fill")+coor
d_flip()
```



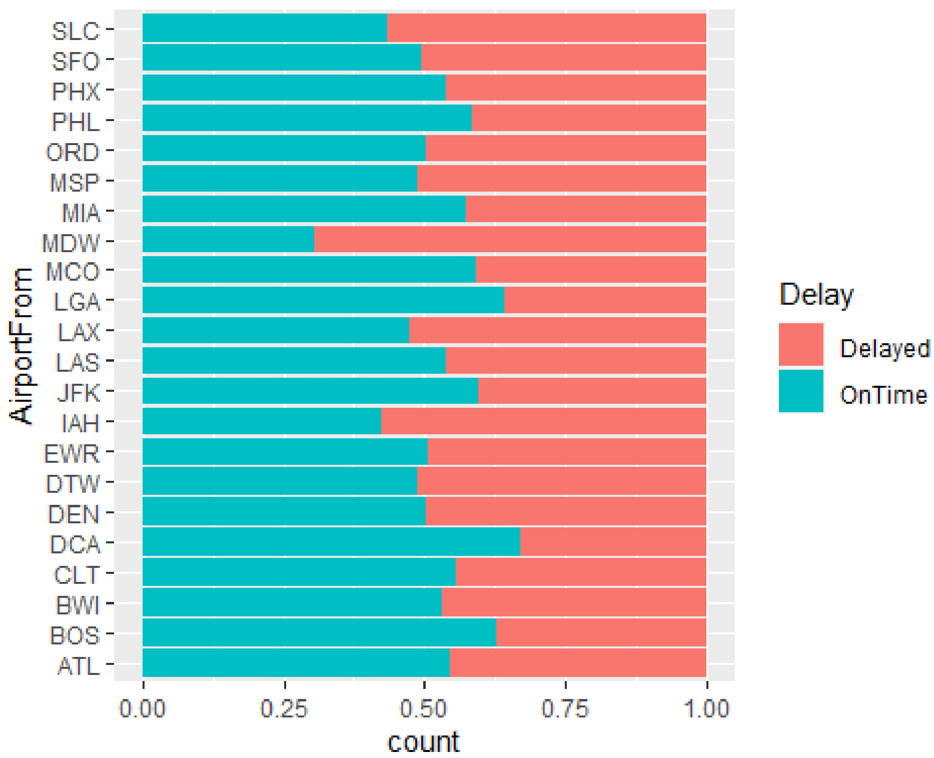

```
#normalized bar graph of DayOfWeek with delay overlay
ggplot(df,aes(DayOfWeek))+geom_bar(aes(fill=Delay),position="fill")+co
ord_flip()
```



```
#normalized bar graph of AirportTo with delay overlay
ggplot(df,aes(AirportTo))+geom_bar(aes(fill=Delay),position="fill")+co
ord_flip()
```




```
#normalized bar graph of AirlineFrom with delay overlay
ggplot(df,aes(AirportFrom))+geom_bar(aes(fill=Delay),position="fill")+
coord_flip()
```




```
#split data set to train and test(75/25)
set.seed(100)
n <- dim(df) [1]
train_ind <- runif(n) < 0.75
df_train <- df[ train_ind, ]
df_test <- df[ !train_ind, ]

dim(df_test)

## [1] 30694     9

#contingency table airline in train data set
cont_tb <- table(df_train$Delay,df_train$Airline)
cont_tb_col <- addmargins(A=cont_tb,FUN=list(total=sum),quiet=TRUE)
cont_tb_col

##
##              9E    AA    AS    B6    CO    DL    EV    F9    FL
MQ    OH
##   Delayed   293  3824    24  1568  4672  9740   150   857  1422  17
28   306
##   OnTime    370  4938    22  1728  2812 10529   269  1046  2923  25
45   741
##   total     663  8762    46  3296  7484 20269   419  1903  4345  42
73  1047
```

```
##
##              OO    UA    US    WN    XE    YV total
##   Delayed  1544  3392  4726  8022   961   256 43485
##   OnTime   1310  6355  8749  3202  1270   476 49285
##   total    2854  9747 13475 11224  2231   732 92770
```

```
round(prop.table(cont_tb,margin=2)*100,1)
```

```
##
##             9E   AA   AS   B6   CO   DL   EV   F9   FL   MQ   OH
OO   UA   US
##   Delayed 44.2 43.6 52.2 47.6 62.4 48.1 35.8 45.0 32.7 40.4 29.2 54
.1 34.8 35.1
##   OnTime  55.8 56.4 47.8 52.4 37.6 51.9 64.2 55.0 67.3 59.6 70.8 45
.9 65.2 64.9
##
##             WN   XE   YV
##   Delayed 71.5 43.1 35.0
##   OnTime  28.5 56.9 65.0
```

```
cont_tb_row <- table(df_train$Airline,df_train$Delay)
cont_tb_row_col <- addmargins(A=cont_tb,FUN=list(total=sum),quiet=TRUE
)
cont_tb_row_col
```

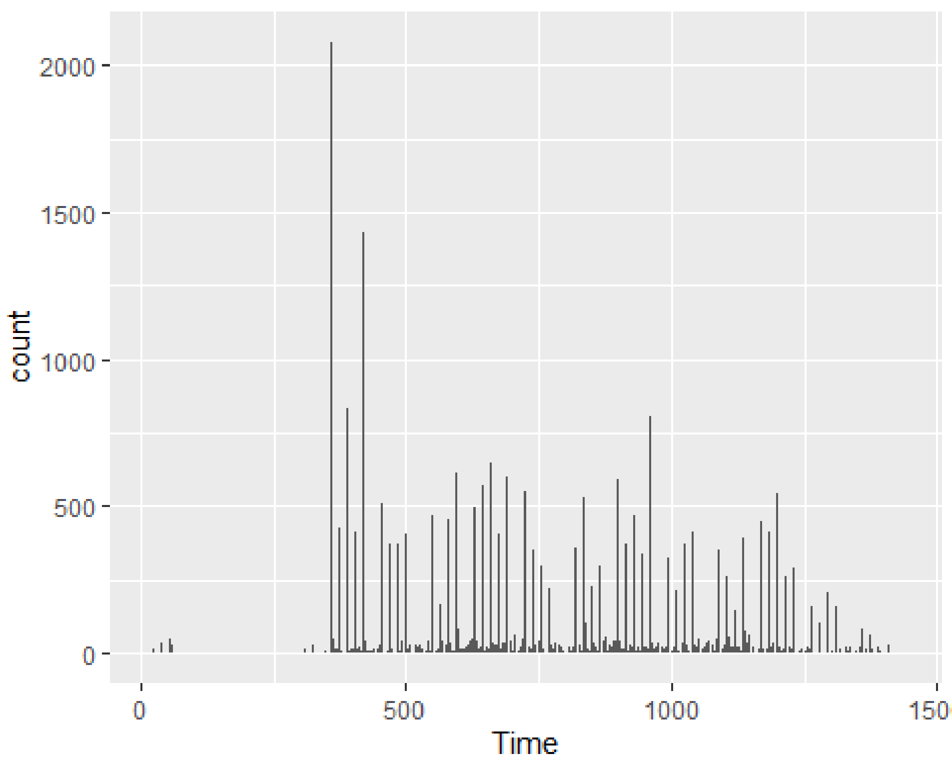
```
##
##              9E    AA    AS    B6    CO    DL    EV    F9    FL
MQ    OH
##   Delayed   293  3824    24  1568  4672  9740   150   857  1422  17
28   306
##   OnTime    370  4938    22  1728  2812 10529   269  1046  2923  25
45   741
##   total     663  8762    46  3296  7484 20269   419  1903  4345  42
73  1047
##
##              OO    UA    US    WN    XE    YV total
##   Delayed  1544  3392  4726  8022   961   256 43485
##   OnTime   1310  6355  8749  3202  1270   476 49285
##   total    2854  9747 13475 11224  2231   732 92770
```

```
round(prop.table(cont_tb_row,margin=2)*100,1)
```

```
##
##      Delayed OnTime
##   9E     0.7    0.8
##   AA     8.8   10.0
##   AS     0.1    0.0
```

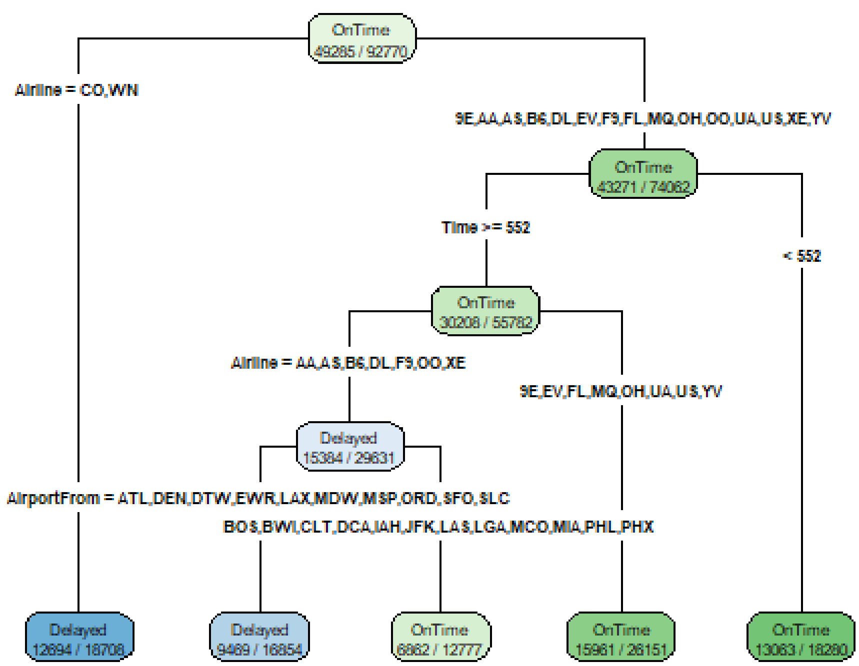
```
##    B6      3.6     3.5
##    CO     10.7     5.7
##    DL     22.4    21.4
##    EV      0.3     0.5
##    F9      2.0     2.1
##    FL      3.3     5.9
##    MQ      4.0     5.2
##    OH      0.7     1.5
##    OO      3.6     2.7
##    UA      7.8    12.9
##    US     10.9    17.8
##    WN     18.4     6.5
##    XE      2.2     2.6
##    YV      0.6     1.0
```

```r
#histogram of Time

ggplot(df_train,aes(Time))+geom_bar()
```



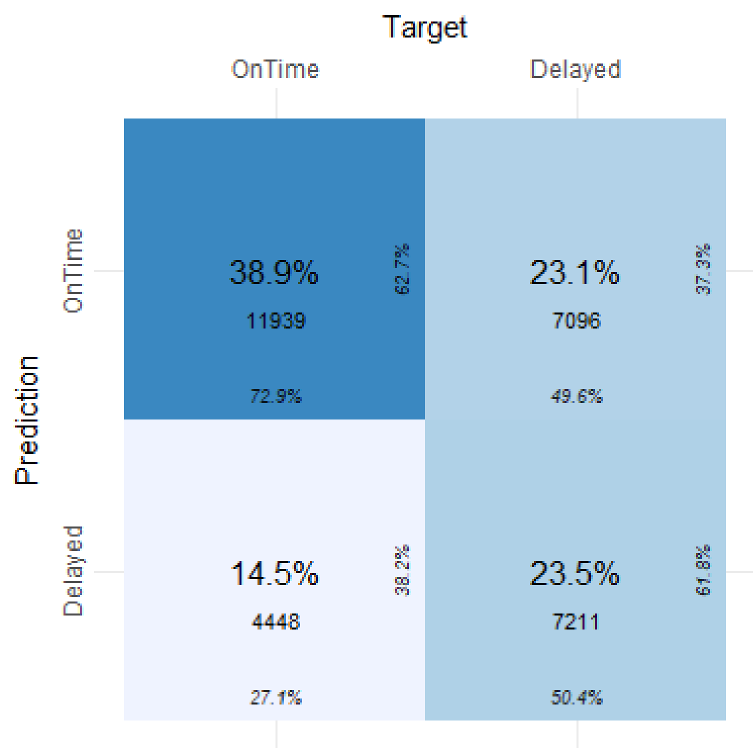

```r
#CART
#install.packages("cvms")
library(cvms)

## Warning: package 'cvms' was built under R version 4.1.3
```

```
library(tibble)
library(rpart); library(rpart.plot)

## Warning: package 'rpart' was built under R version 4.1.3

## Warning: package 'rpart.plot' was built under R version 4.1.3

df_train$Delay <- factor(df_train$Delay)
df_train$DayOfWeek <- factor(df_train$DayOfWeek)
df_train$AirportTo <- factor(df_train$AirportTo)
df_train$AirportFrom <- factor(df_train$AirportFrom)
cart01 <- rpart(formula = Delay ~ Airline+Time+AirportFrom+DayOfWeek,
data = df_train, method = "class")
rpart.plot(cart01,type=4,extra=2)
```



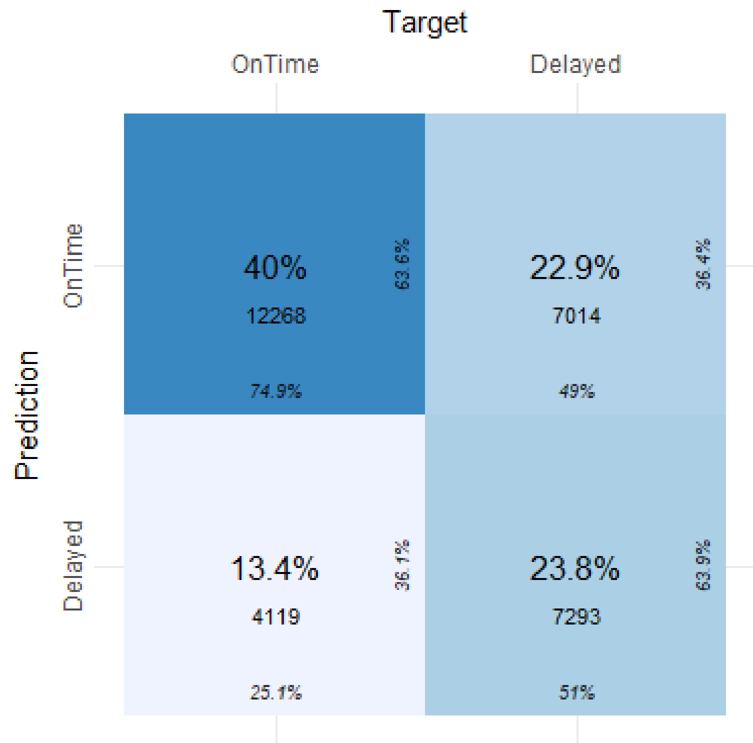

```
cart01$variable.importance

##     Airline        Time  AirportFrom
##   2560.4647    831.7494    538.6659

df_test$Delay <- factor(df_test$Delay)
df_test$DayOfWeek <- factor(df_test$DayOfWeek)
df_train$AirportTo <- factor(df_train$AirportTo)
df_train$Airportfrom <- factor(df_train$AirportFrom)

X = data.frame(Airline=df_test$Airline,Time=df_test$Time,AirportFrom=d
```

```
f_test$AirportFrom,AirportTo=df_test$AirportTo,Length=df_test$Length,F
light=df_test$Flight,DayOfWeek=df_test$DayOfWeek)

predDelayCART = predict(object = cart01, newdata = X,type = "class")
t1=table(df_test$Delay, predDelayCART)
t1

##          predDelayCART
##           Delayed OnTime
##   Delayed    7211   7096
##   OnTime     4448  11939

#Confusion Matrix CART
cfmcart <- as_tibble(t1,  .name_repair = ~c("target","prediction","n")
)
plot_confusion_matrix(cfmcart,
                      target_col = "target",
                      prediction_col = "prediction",
                      counts_col = "n")

## Warning in plot_confusion_matrix(cfmcart, target_col = "target", pr
ediction_col
## = "prediction", : 'ggimage' is missing. Will not plot arrows and ze
ro-shading.

## Warning in plot_confusion_matrix(cfmcart, target_col = "target", pr
ediction_col
## = "prediction", : 'rsvg' is missing. Will not plot arrows and zero-
shading.
```




```
accuracycart = (t1[1,1]+t1[2,2])/nrow(df_test)
accuracycart

## [1] 0.6239004

Precissioncart = t1[1,1]/(t1[1,1]+t1[1,2])
Precissioncart

## [1] 0.504019

Recallcart = t1[1,1]/(t1[1,1]+t1[2,1])
Recallcart

## [1] 0.6184922

F1Scorecart = 2*Precissioncart*Recallcart/(Precissioncart+Recallcart)
F1Scorecart

## [1] 0.5554186

#C5
library(C50)

## Warning: package 'C50' was built under R version 4.1.3
```

```
C5 <- C5.0(formula = Delay ~ Airline+Time+AirportFrom+DayOfWeek, data
=df_train, control = C5.0Control(minCases=50))
C5

##
## Call:
## C5.0.formula(formula = Delay ~ Airline + Time + AirportFrom + DayOf
Week, data
##  = df_train, control = C5.0Control(minCases = 50))
##
## Classification Tree
## Number of samples: 92770
## Number of predictors: 4
##
## Tree size: 115
##
## Non-standard options: attempt to group attributes, minimum number o
f cases: 50

#plot(C5)
predDelayC5=predict(object = C5, newdata = X)
t2=table(df_test$Delay, predDelayC5)
t2

##          predDelayC5
##           Delayed OnTime
##   Delayed    7293   7014
##   OnTime     4119  12268

#Confusion Matrix C5.0
cfmc5 <- as_tibble(t2,  .name_repair = ~c("target","prediction","n"))
plot_confusion_matrix(cfmc5,
                      target_col = "target",
                      prediction_col = "prediction",
                      counts_col = "n")

## Warning in plot_confusion_matrix(cfmc5, target_col = "target", pred
iction_col =
## "prediction", : 'ggimage' is missing. Will not plot arrows and zero
-shading.

## Warning in plot_confusion_matrix(cfmc5, target_col = "target", pred
iction_col =
## "prediction", : 'rsvg' is missing. Will not plot arrows and zero-sh
ading.
```




```
accuracyC5 = (t2[1,1]+t2[2,2])/nrow(df_test)
accuracyC5

## [1] 0.6372907

PrecissionC5 = t2[1,1]/(t2[1,1]+t2[1,2])
PrecissionC5

## [1] 0.5097505

RecallC5 = t2[1,1]/(t2[1,1]+t2[2,1])
RecallC5

## [1] 0.6390641

F1ScoreC5 = 2*PrecissionC5*RecallC5/(PrecissionC5+RecallC5)
F1ScoreC5

## [1] 0.5671294

#Randomforest
library(randomForest)

## Warning: package 'randomForest' was built under R version 4.1.3

## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine

rf01 <- randomForest(formula = Delay ~ Airline+Time+AirportFrom+DayOfW
eek, data =df_train, ntree = 100, type = 'classification')
test.rf <- subset(x=df_train,select=c("Airline","AirportFrom","Time","
DayOfWeek"))
rf_pred <- predict(object = rf01, newdata = test.rf)
rf_table <- table(df_train$Delay,rf_pred)
rf_table

##          rf_pred
##           Delayed OnTime
##   Delayed   29084  14401
##   OnTime     9696  39589

row.names(rf_table) <- c("Delayed","Ontime")

colnames (rf_table) <- c("Delayed","Ontime")
rf_table <- addmargins(A = rf_table, FUN = list(Total = sum), quiet =
TRUE)
rf_table

##          rf_pred
##           Delayed Ontime Total
##   Delayed   29084  14401 43485
##   Ontime     9696  39589 49285
##   Total     38780  53990 92770

#install.packages("OneR")
library(OneR)

## Warning: package 'OneR' was built under R version 4.1.3

eval_model(df_train$Delay, rf_pred)

##
## Confusion matrix (absolute):
##             Actual
```

```
## Prediction Delayed OnTime   Sum
##    Delayed   29084  14401 43485
##    OnTime     9696  39589 49285
##    Sum       38780  53990 92770
##
## Confusion matrix (relative):
##             Actual
## Prediction Delayed OnTime  Sum
##    Delayed    0.31   0.16 0.47
##    OnTime     0.10   0.43 0.53
##    Sum        0.42   0.58 1.00
##
## Accuracy:
## 0.7403 (68673/92770)
##
## Error rate:
## 0.2597 (24097/92770)
##
## Error rate reduction (vs. base rate):
## 0.3786 (p-value < 2.2e-16)
```

```
#check for test data in Random forest
rf02 <- randomForest(formula = Delay ~ Airline+Time+AirportFrom+DayOfW
eek, data =df_test, ntree = 100,
                     type = 'classification')
test.rf2 <- subset(x=df_test,select=c("Airline","AirportFrom","Time","
DayOfWeek"))
rf_pred2 <- predict(object = rf02, newdata = test.rf2)
rf_table2 <- table(df_test$Delay,rf_pred2)
rf_table2
```

```
##          rf_pred2
##           Delayed OnTime
##   Delayed   11716   2591
##   OnTime     2730  13657
```

```
row.names(rf_table2) <- c("Delayed","Ontime")

colnames (rf_table2) <- c("Delayed","Ontime")
rf_table2 <- addmargins(A = rf_table2, FUN = list(Total = sum), quiet
= TRUE)
rf_table2
```

```
##          rf_pred2
##           Delayed Ontime Total
##   Delayed   11716   2591 14307
```

```
##   Ontime      2730  13657 16387
##   Total      14446  16248 30694
```

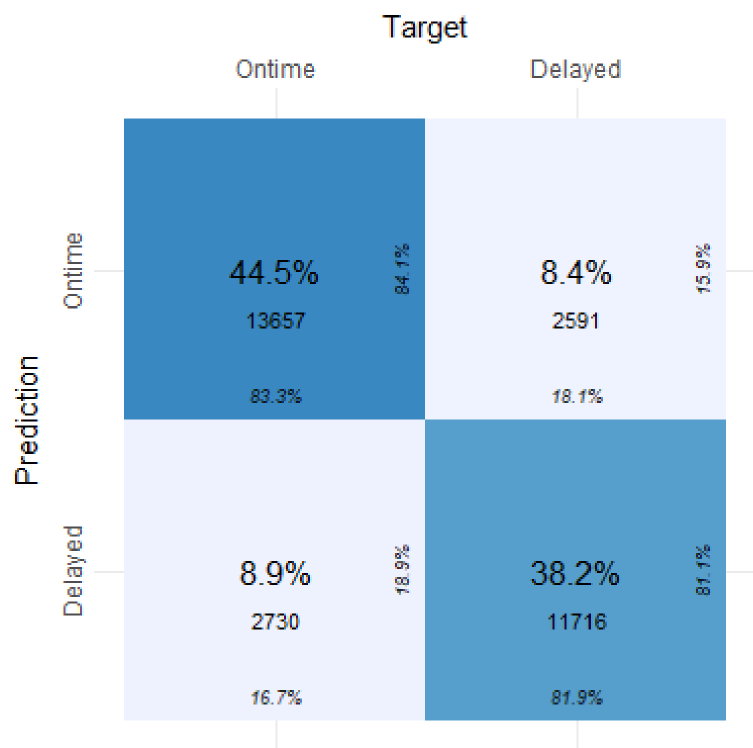
```
eval_model(df_test$Delay, rf_pred2)
```

```
##
## Confusion matrix (absolute):
##            Actual
## Prediction Delayed OnTime   Sum
##    Delayed   11716   2591 14307
##    OnTime     2730  13657 16387
##    Sum       14446  16248 30694
##
## Confusion matrix (relative):
##            Actual
## Prediction Delayed OnTime  Sum
##    Delayed    0.38   0.08 0.47
##    OnTime     0.09   0.44 0.53
##    Sum        0.47   0.53 1.00
##
## Accuracy:
## 0.8266 (25373/30694)
##
## Error rate:
## 0.1734 (5321/30694)
##
## Error rate reduction (vs. base rate):
## 0.6317 (p-value < 2.2e-16)
```

```
#Confusion Matrix Random Forest
cfmrf <- as_tibble(rf_table2[-3,-3],  .name_repair = ~c("target","pred
iction","n"))
plot_confusion_matrix(cfmrf,
                      target_col = "target",
                      prediction_col = "prediction",
                      counts_col = "n")
```

```
## Warning in plot_confusion_matrix(cfmrf, target_col = "target", pred
iction_col =
## "prediction", : 'ggimage' is missing. Will not plot arrows and zero
-shading.
```

```
## Warning in plot_confusion_matrix(cfmrf, target_col = "target", pred
iction_col =
## "prediction", : 'rsvg' is missing. Will not plot arrows and zero-sh
ading.
```




```
accuracyRF = (rf_table2[1,1]+rf_table2[2,2])/nrow(df_test)
accuracyRF

## [1] 0.8266436

PrecissionRF = rf_table2[1,1]/(rf_table2[1,1]+rf_table2[1,2])
PrecissionRF

## [1] 0.8188998

RecallRF = rf_table2[1,1]/(rf_table2[1,1]+rf_table2[2,1])
RecallRF

## [1] 0.8110204

F1ScoreRF = 2*PrecissionRF*RecallRF/(PrecissionRF+RecallRF)
F1ScoreRF

## [1] 0.814941

#Naive bayes
library(e1071)

## Warning: package 'e1071' was built under R version 4.1.3
```

```
nb01 <- naiveBayes(formula = Delay ~ Airline+Time+AirportFrom+DayOfWee
k, data =df_train)
nb01
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##   Delayed    OnTime
## 0.4687399 0.5312601
##
## Conditional probabilities:
##          Airline
## Y                   9E           AA           AS           B6
CO
##   Delayed 0.0067379556 0.0879383696 0.0005519145 0.0360584109 0.107
4393469
##   OnTime  0.0075073552 0.1001927564 0.0004463833 0.0350613777 0.057
0558994
##          Airline
## Y                   DL           EV           F9           FL
MQ
##   Delayed 0.2239852823 0.0034494653 0.0197079453 0.0327009314 0.039
7378406
##   OnTime  0.2136349802 0.0054580501 0.0212234960 0.0593081059 0.051
6384295
##          Airline
## Y                   OH           OO           UA           US
WN
##   Delayed 0.0070369093 0.0355064965 0.0780039094 0.1086811544 0.184
4774060
##   OnTime  0.0150350005 0.0265800954 0.1289438977 0.1775185148 0.064
9690575
##          Airline
## Y                   XE           YV
##   Delayed 0.0220995746 0.0058870875
##   OnTime  0.0257684894 0.0096581110
##
##          Time
## Y             [,1]     [,2]
##   Delayed 844.8585 271.1039
##   OnTime  760.0322 286.2806
```

```
##
##          AirportFrom
## Y                ATL        BOS        BWI        CLT        DCA
DEN
##   Delayed 0.08090146 0.04470507 0.03093021 0.04316431 0.02492814 0.
06347016
##   OnTime  0.08509689 0.06632850 0.03069900 0.04766156 0.04640357 0.
05543269
##          AirportFrom
## Y                DTW        EWR        IAH        JFK        LAS
LAX
##   Delayed 0.04415316 0.03936990 0.05020122 0.03230999 0.04553294 0.
06977119
##   OnTime  0.03656285 0.03546718 0.03211931 0.04214264 0.04709344 0.
05492543
##          AirportFrom
## Y                LGA        MCO        MDW        MIA        MSP
ORD
##   Delayed 0.03286191 0.04054272 0.03677130 0.03109118 0.04134759 0.
07906175
##   OnTime  0.05147611 0.05151669 0.01369585 0.03650198 0.03619763 0.
07077204
##          AirportFrom
## Y                PHL        PHX        SFO        SLC
##   Delayed 0.03295389 0.05043118 0.04815454 0.03734621
##   OnTime  0.03974840 0.05218626 0.04185858 0.02611342
##
##          DayOfWeek
## Y            Friday    Monday  Thursday   Tuesday Wednesday   Weeke
nd
##   Delayed 0.1502817 0.1362999 0.1723813 0.1331034 0.1793952 0.22853
86
##   OnTime  0.1701329 0.1335701 0.1668459 0.1286395 0.1533124 0.24749
92
```

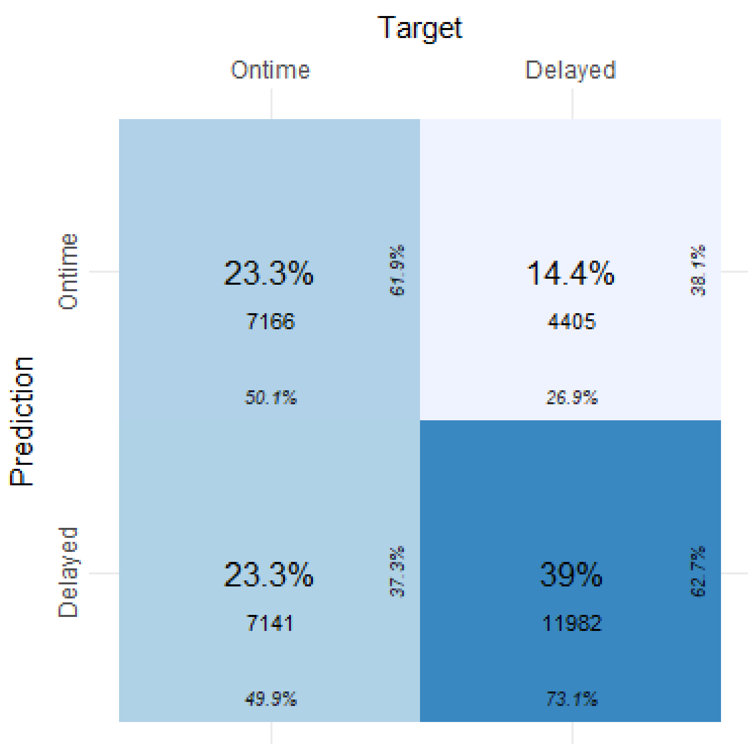
```
ypred <- predict(object=nb01,newdata=df_test)
#Create contingency table of actual vs. predicted values
t.preds <- table(df_test$Delay, ypred)
rownames(t.preds) <- c("Ontime", "Delayed")
colnames(t.preds) <- c("Ontime", "Delayed")
addmargins(A = t.preds, FUN = list(Total = sum), quiet = TRUE)
```

```
##          ypred
##           Ontime Delayed Total
##   Ontime    7166    7141 14307
```

```
##    Delayed    4405    11982 16387
##    Total     11571    19123 30694
```

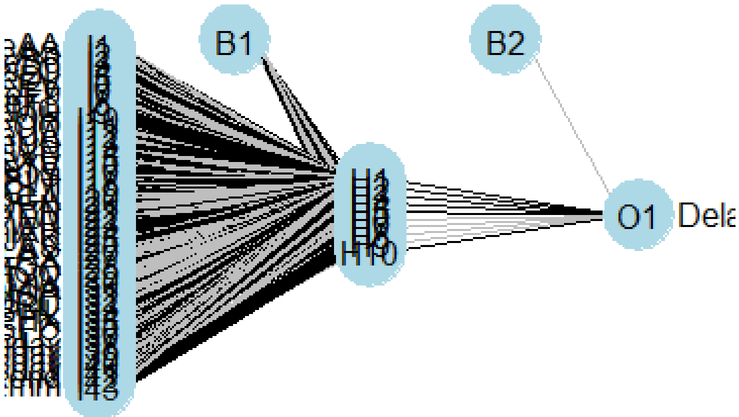
```
#Confusion Matrix Naive Bayes
cfmnb <- as_tibble(t.preds,  .name_repair = ~c("target","prediction","
n"))
plot_confusion_matrix(cfmnb,
                      target_col = "target",
                      prediction_col = "prediction",
                      counts_col = "n")
```

```
## Warning in plot_confusion_matrix(cfmnb, target_col = "target", pred
iction_col =
## "prediction", : 'ggimage' is missing. Will not plot arrows and zero
-shading.

## Warning in plot_confusion_matrix(cfmnb, target_col = "target", pred
iction_col =
## "prediction", : 'rsvg' is missing. Will not plot arrows and zero-sh
ading.
```



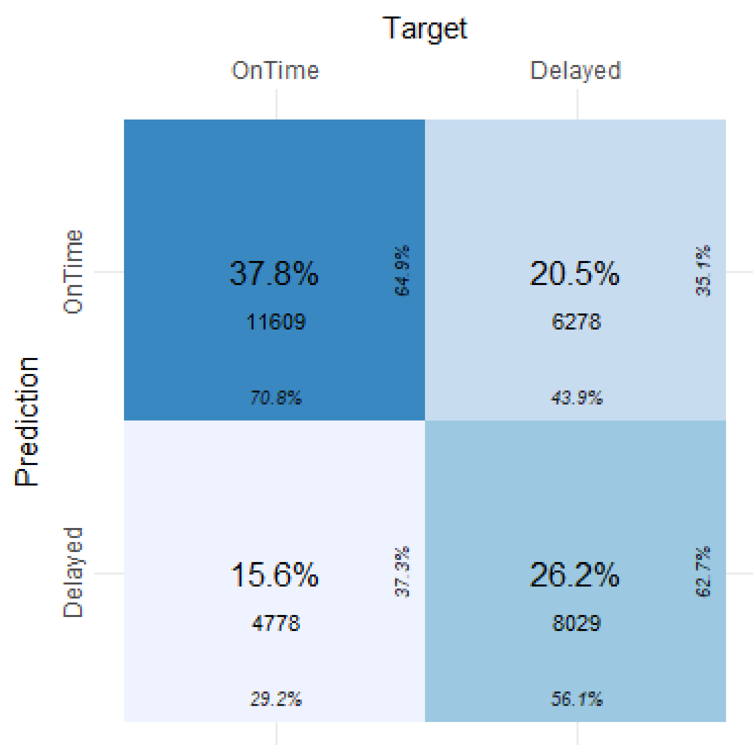

```
accuracyNB = (t.preds[1,1]+t.preds[2,2])/nrow(df_test)
accuracyNB

## [1] 0.6238353
```

```
PrecissionNB = t.preds[1,1]/(t.preds[1,1]+t.preds[1,2])
PrecissionNB

## [1] 0.5008737

RecallNB = t.preds[1,1]/(t.preds[1,1]+t.preds[2,1])
RecallNB

## [1] 0.6193069

F1ScoreNB = 2*PrecissionNB*RecallNB/(PrecissionNB+RecallNB)
F1ScoreNB

## [1] 0.5538295

#Neural Network
df_train$Time.mm <- (df_train$Time - min(df_train$Time)) /(max(df_trai
n$Time) - min(df_train$Time))
library(nnet)

## Warning: package 'nnet' was built under R version 4.1.3

library(NeuralNetTools)

## Warning: package 'NeuralNetTools' was built under R version 4.1.3

neunet_01 <- nnet(Delay ~ Airline+AirportFrom+DayOfWeek + Time.mm, dat
a = df_train, size = 10)

## # weights:  451
## initial  value 70745.752815
## iter  10 value 59693.527260
## iter  20 value 59088.088290
## iter  30 value 58734.896630
## iter  40 value 58482.421933
## iter  50 value 58326.942956
## iter  60 value 58163.823358
## iter  70 value 58004.693593
## iter  80 value 57923.278002
## iter  90 value 57864.147003
## iter 100 value 57816.653485
## final  value 57816.653485
## stopped after 100 iterations

X_train <- subset(x=df_train, select =c("Time.mm", "Airline", "Airport
From", "Delay","DayOfWeek"))
ypred <- predict(object = neunet_01, newdata = X_train)
ypred <- ifelse(ypred > 0.5, yes="Delayed", no="OnTime")
plotnet(neunet_01)
```




```
#Evaluate neural network
df_test$Time.mm <- (df_test$Time - min(df_test$Time)) /(max(df_test$Ti
me) - min(df_test$Time))
X_test <- subset(x=df_test, select =c("Time.mm", "Airline", "AirportFr
om", "Delay","DayOfWeek"))
ypred <- predict(object = neunet_01, newdata = X_test)
ypred <- ifelse(ypred > 0.5, yes="OnTime", no="Delayed")
table1 <- table(df_test$Delay, ypred)
table1 <- addmargins(A=table1, FUN=list(Total=sum), quiet = TRUE)
table1

##          ypred
##           Delayed OnTime Total
##   Delayed    8029   6278 14307
##   OnTime     4778  11609 16387
##   Total     12807  17887 30694

#Confusion Matrix Neural Network
cfmnb <- as_tibble(table1[-3,-3],  .name_repair = ~c("target","predict
ion","n"))
plot_confusion_matrix(cfmnb,
                      target_col = "target",
                      prediction_col = "prediction",
                      counts_col = "n")
```

```
## Warning in plot_confusion_matrix(cfmnb, target_col = "target", pred
iction_col =
## "prediction", : 'ggimage' is missing. Will not plot arrows and zero
-shading.

## Warning in plot_confusion_matrix(cfmnb, target_col = "target", pred
iction_col =
## "prediction", : 'rsvg' is missing. Will not plot arrows and zero-sh
ading.
```




```
accuracyNN = (table1[1,1]+table1[2,2])/nrow(df_test)
accuracyNN

## [1] 0.6397993

PrecissionNN = table1[1,1]/(table1[1,1]+table1[1,2])
PrecissionNN

## [1] 0.5611938

RecallNN = table1[1,1]/(table1[1,1]+table1[2,1])
RecallNN

## [1] 0.6269228
```

```
F1ScoreNN = 2*PrecissionNN*RecallNN/(PrecissionNN+RecallNN)
F1ScoreNN

## [1] 0.5922402
```