Final Project

Aditya, Sumeer and Rohan

Sunday, June 28, 2015

Contents

Introduction

Time is a critical criterion for the airlines industry. Having planes scheduled in timely manner is the yardstick of success for this industry. Everyday, there are thousands of people who travel by airplane to get to their destinations. Unfortunately for airline travelers, however, many of these flights do not leave

on-time.Air travel cancellation has been a universal problem. As more and more economic connections happen among different countries, this issue can cause huge problems to frequent travellers, especially long-distance travellers.

Flight delays are a major problem within the airline industry. The project explores what factors influence the occurrence of flight delays. Our research acumen focusses on generating some interesting insights regarding air travel cancellation. On acknowledging our insights, travellers can plan ahead accordingly, airline carriers and airports can make efforts to reduce cancellation based on our findings, and hotels can plan their marketing and sales based on certain flight cancellation pattern.

Background

The available dataset is provided by Research and Innovative Technology Administration (RITA) and Bureau of Transportation Statistics (BTS). The available data includes arrival and departure data for 123 million commercial flights throughout the United States. The corresponding dataset gave provision for on-time arrival data for non-stop domestic flights by major air carriers. This data-set possesses detailed facets of airline information from October 1987 to December 2008.

Some interesting statistics were found about the Airlines dataset. The airline industry is responsible for incurring an average cost of about $11,300 per delayed flight. This figure is based on 61,000 delayed flights on a per month average and excludes costs to passengers and lost demand.

For our assignment, we have narrowed down our investigations to data recorded for the year of 2008.

There are a total of 29 variables for each record. The variables are described in detail in the **Dataset Description** section.

Goal

Flight delay is a challenging problem that needs to be addressed, because, this can lead to financial losses or harm business reputation. There are over 300,000 flights within the United States that fly every day. Nobody wants to gamble by choosing a flight that will have a high likelihood to be late.

By utilizing the graphical techniques, we analyze the 2008 Expo data set to discover some patterns about the delayed flights according to time, locations,

and different carriers, amongst other factors. We try to show a reasonable way to pick a flight based on different criteria. Visualizing the airline on-time performance can give a traveler accurate information in making a smart decision on booking a flight. This in return also motivates airlines and airports to find efficient ways to improve their performance.The purpose of our questions is aimed to find a better way for identifying best flights and avoiding the worst airports.

The goal is to reproduce factors associated with flights delays. This is developed by analyzing the trend with respect to arrival and departure delays. The models are then integrated in the form of a system for aircraft delay analysis and airport delay assessment. In this manner, we can concentrate on investigating the delays at the flight level.

Dataset description

The data set consists of following variables

1. `Year` 1987-2008
2. `Month` 1-12
3. `DayofMonth` 1-31
4. `DayOfWeek` 1 (Monday) - 7 (Sunday)
5. `DepTime` actual departure time (local, hhmm)
6. `CRSDepTime` scheduled departure time (local, hhmm)
7. `ArrTime` actual arrival time (local, hhmm)
8. `CRSArrTime` scheduled arrival time (local, hhmm)
9. `UniqueCarrier` unique carrier code
10. `FlightNum` flight number
11. `TailNum` plane tail number
12. `ActualElapsedTime` in minutes
13. `CRSElapsedTime` in minutes
14. `AirTime` in minutes
15. `ArrDelay` arrival delay, in minutes
16. `DepDelay` departure delay, in minutes
17. `Origin` origin IATA airport code
18. `Dest` destination IATA airport code
19. `Distance` in miles
20. `TaxiIn` taxi in time, in minutes

21. `TaxiOut` taxi out time in minutes
22. `Cancelled` was the flight cancelled?
23. `CancellationCode` reason for cancellation (A = carrier, B = weather, C = NAS, D = security)
24. `Diverted` 1 = yes, 0 = no
25. `CarrierDelay` in minutes
26. `WeatherDelay` in minutes
27. `NASDelay` in minutes
28. `SecurityDelay` in minutes
29. `LateAircraftDelay` in minutes

In a nutshell, the dataset comprises of on-time arrival data for non-stop domestic flights by major air carriers, and provides such additional items as departure and arrival delays, origin and destination airports, flight numbers, scheduled and actual departure and arrival times, cancelled or diverted flights, taxi-out and taxi-in times, air time, and non-stop distance.

Data preparation

Once the working directory was set, the file was downloaded and extracted after which the csv file was then imported using:

```
library(data.table)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:data.table':
##
##     between, last
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following objects are masked from 'package:base':
```

```
##
##     intersect, setdiff, setequal, union
```

```
sample <- fread("2008.csv")
```

```
## Warning in fread("2008.csv"): Bumped column 23 to type character on data
## row 179, field contains 'A'. Coercing previously read values in this
## column from logical, integer or numeric back to character which may not
## be lossless; e.g., if '00' and '000' occurred before they will now be just
## '0', and there may be inconsistencies with treatment of ',,' and ',NA,' too
## (if they occurred in this column before the bump). If this matters please
## rerun and set 'colClasses' to 'character' for this column. Please note
## that column type detection uses the first 5 rows, the middle 5 rows and the
## last 5 rows, so hopefully this message should be very rare. If reporting to
## datatable-help, please rerun and include the output from verbose=TRUE.
```

```
##
Read 5.3% of 7009728 rows
Read 24.1% of 7009728 rows
Read 42.9% of 7009728 rows
Read 61.6% of 7009728 rows
Read 80.0% of 7009728 rows
Read 98.0% of 7009728 rows
Read 7009728 rows and 29 (of 29) columns from 0.642 GB file in 00:00:08
```

We discard records with the NA values to have a cleaner dataset.

```
sample = na.omit(sample)
```

Variables Information in depth

The variables are assessed further for detailed information on the subject matter.

1. `DayofMonth` : day of May (1-31)

```
summary(sample$DayofMonth)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     8.0    16.0    15.7    23.0    31.0
```

2. `DayOfWeek`: Looking at the dataset, we notice that the 'DayOfWeek' is represented by a numeric value. Therefore, We will be re assigning these numeric values to three letter day abbreviations. For e.g: Sun through Mon (Represented as 1-7).

```
summary(sample$DayOfWeek)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   4.000   3.964   6.000   7.000
```

3. `DepTime ArrTime` : 'DepTime' is the actual departure time 'ArrTime' is actual arrival time. The best time for a flight is a key performance indicator for analysis.

```
summary(sample$DepTime)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1    1140    1541    1502    1902    2400
```

```
summary(sample$ArrTime)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1    1249    1711    1590    2034    2400
```

4.`CRSDepTime, CRSArrTime` : Scheduled departure and arrival time It is divided into hours and minutes.

```
summary(sample$CRSDepTime)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0    1110    1500    1445    1813    2359
```

```
summary(sample$CRSArrTime)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0    1256    1656    1615    2010    2359
```

5. `UniqueCarrier` : Two letter abbreviation for airline carrier

```
carrierDesc = data.frame(sample$UniqueCarrier)
```

6. `FlightNum`: This indicates the Flight Number for a carrier

```
summary(sample$FlightNum)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1     625    1560    2241    3600    9741
```

7. `Actual Elapsed Time, CRSElapsed time` (time in minutes)

```
summary(sample$ActualElapsedTime)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    14.0    88.0   126.0   143.9   178.0  1379.0
```

```
summary(sample$CRSElapsedTime)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -141.0    83.0   118.0   135.6   165.0   660.0
```

8. `AirTime` (time in minutes)

```
summary(sample$AirTime)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    60.0    94.0   112.9   143.0  1350.0
```

9. `ArrDelay and DepDelay` Actual arrival delay in minutes (early departures are counted as 0) Actual departure delay in minutes

```r
summary(sample$ArrDelay)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   15.00   23.00   37.00   56.82   69.00 2461.00
```

```r
summary(sample$DepDelay)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -61.00   13.00   32.00   48.54   65.00 2467.00
```

10. `Origin` : Aircraft aiport code

```r
summary(sample$Origin)
```

```
##    Length     Class      Mode
##   1524735 character character
```

11. `Dest` : Destination airport code

```r
summary(sample$Dest)
```

```
##    Length     Class      Mode
##   1524735 character character
```

12. `Distance`: Distance in miles the flight must travel after its deprture

```r
summary(sample$Distance)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    11.0   337.0   610.0   769.9  1005.0  4962.0
```

The proportionality between distance and delay time is an important relationship for analysis.

Exploratory Analysis

Do all airlines follow Hub-Spoke system? Hub-Spoke system is a transport system in which passengers travel from smaller airports to one large central airport to make longer trips. A major disadvantage of hub-spoke system is that any disruption at hub can create major delays. As travellers, we would want to know hubs for various airlines so that we can make an informed choice for chances of delay depending on delay at hubs that we might travel to. We will visualize the flight path using spatial information from `airports.csv`.

Lets import the libraries and read our datasets,

```
library(dplyr)
library(data.table)
flightInfo <- sample
airports <- read.csv("airports.csv", header = TRUE)
```

After loading the required libraries and our dataset as shown above, lets find how many unique carriers are present in our dataset.

```
#filter nodes for each airline
carriers <- flightInfo %>% select(UniqueCarrier) %>% distinct(UniqueCarrier)
na.omit(carriers)
```

We make use of `maps` and `geosphere` libraries for geospatial plot. `geosphere` gives us `gcintermediate()` function which helps us draw circular lines depicting flight path.

```
library(maps)
library(geosphere)
```

```
## Warning: package 'geosphere' was built under R version 3.2.1
```

```
## Loading required package: sp
```

We limit the latitude and longitude co-ordinates to display a map of USA along with US islands. We generate a pallete of colors of a particular shade with various shades. More brighter shades are used to depict more frequency of flights.

```
xlim <- c(-171.738281,-56.601563)
ylim <- c(12.039321, 71.856229)
pal <- colorRampPalette(c("#333333", "white", "#1292db"))
colors <- pal(100)
```

Lets look at the flight pattern for carrier `WN`

```
flightInfo.org_dest.WN <- flightInfo %>%
  filter(UniqueCarrier == 'WN') %>%
  select(FlightNum, Origin, Dest, UniqueCarrier) %>%
  distinct(FlightNum, Origin, Dest) %>%
  group_by(Origin,Dest) %>%
  summarize(cnt=n())
map("world", col="#191919", fill=T, bg="#333300", lwd=0.05, xlim=xlim, ylim=ylim)
flightInfo.org_dest.WN<-flightInfo.org_dest.WN[order(flightInfo.org_dest.WN$cnt),]
maxcnt <- max(flightInfo.org_dest.WN$cnt)
for (j in 1:length(flightInfo.org_dest.WN$Origin)) {
  air1 <-
    airports[airports$iata == as.character(flightInfo.org_dest.WN[j,]$Origin),]
  air2 <-
    airports[airports$iata == as.character(flightInfo.org_dest.WN[j,]$Dest),]

  inter <- gcIntermediate(c(air1[1,]$long, air1[1,]$lat),
                          c(air2[1,]$long, air2[1,]$lat),
                          n = 100,
                          addStartEnd = TRUE
  )
  colindex <- round( (flightInfo.org_dest.WN[j,]$cnt / maxcnt) * length(colors))
  lines(inter, col = colors[colindex], lwd = 0.6)
}
```

From the plot we can see that `WN` i.e. Southwest airlines has many busy airports but the spokes are not quite prominent to classify some areas as hubs.

Looking at flight pattern for carrier `XE` i.e Express Jet

```
#Network for XE on map
#XE
flightInfo.org_dest.XE <- flightInfo %>%
  filter(UniqueCarrier == 'XE') %>%
  select(FlightNum, Origin, Dest, UniqueCarrier) %>%
  distinct(FlightNum, Origin, Dest) %>%
  group_by(Origin,Dest) %>%
  summarize(cnt=n())

map("world", col="#191919", fill=TRUE, bg="#333300", lwd=0.05, xlim=xlim, ylim=yli
flightInfo.org_dest.XE<-flightInfo.org_dest.XE[order(flightInfo.org_dest.XE$cnt),]
maxcnt <- max(flightInfo.org_dest.XE$cnt)
for (j in 1:length(flightInfo.org_dest.XE$Origin)) {
  air1 <-
    airports[airports$iata == as.character(flightInfo.org_dest.XE[j,]$Origin),]
  air2 <-
```

```
    airports[airports$iata == as.character(flightInfo.org_dest.XE[j,]$Dest),]

  inter <- gcIntermediate(c(air1[1,]$long, air1[1,]$lat),
                          c(air2[1,]$long, air2[1,]$lat),
                          n = 100,
                          addStartEnd = TRUE
  )
  colindex <- round( (flightInfo.org_dest.XE[j,]$cnt / maxcnt) * length(colors))
  lines(inter, col = colors[colindex], lwd = 0.6)
}
```

We observe atleast one hub here in south.

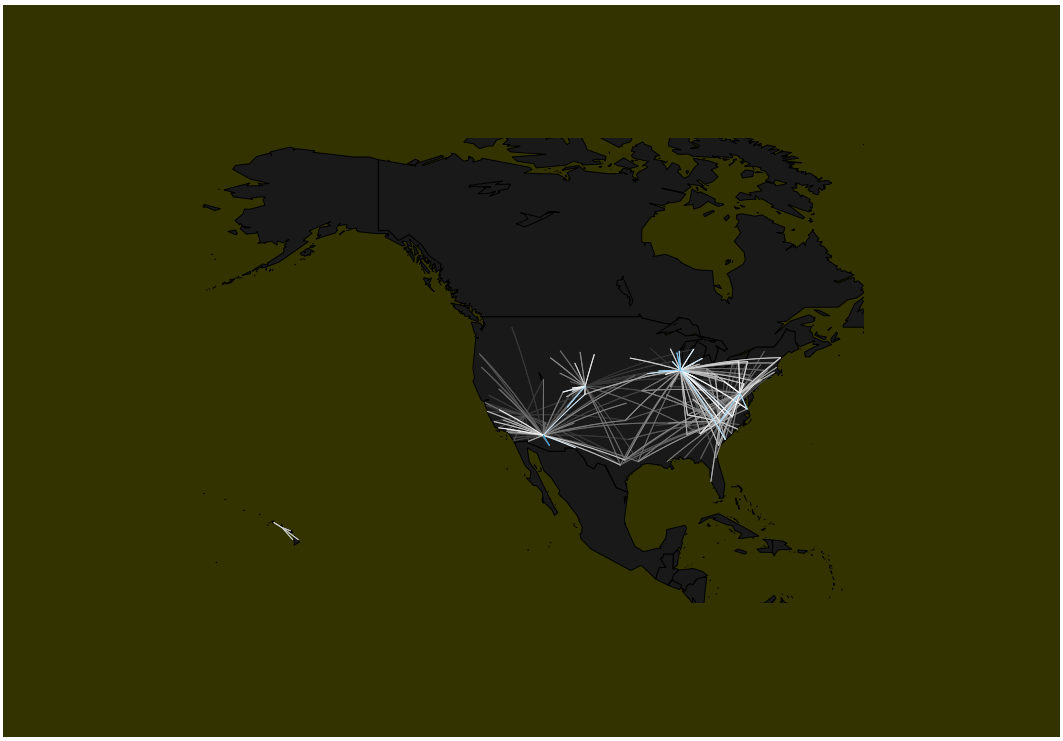
Similarly for flight pattern for carrier 'YV',

```
#Network for YV on map
#YV
flightInfo.org_dest.YV <- flightInfo %>%
  filter(UniqueCarrier == 'YV') %>%
  select(FlightNum, Origin, Dest, UniqueCarrier) %>%
```

```
  distinct(FlightNum, Origin, Dest) %>%
  group_by(Origin,Dest) %>%
  summarize(cnt=n())

map("world", col="#191919", fill=TRUE, bg="#333300", lwd=0.05, xlim=xlim, ylim=yli
flightInfo.org_dest.YV<-flightInfo.org_dest.YV[order(flightInfo.org_dest.YV$cnt),]
maxcnt <- max(flightInfo.org_dest.YV$cnt)
for (j in 1:length(flightInfo.org_dest.YV$Origin)) {
  air1 <-
    airports[airports$iata == as.character(flightInfo.org_dest.YV[j,]$Origin),]
  air2 <-
    airports[airports$iata == as.character(flightInfo.org_dest.YV[j,]$Dest),]

  inter <- gcIntermediate(c(air1[1,]$long, air1[1,]$lat),
                          c(air2[1,]$long, air2[1,]$lat),
                          n = 100,
                          addStartEnd = TRUE
  )
  colindex <- round( (flightInfo.org_dest.YV[j,]$cnt / maxcnt) * length(colors))
  lines(inter, col = colors[colindex], lwd = 0.6)
}
```



Lets see one example of small carrier like `EV`

```
#Network for EV on map
#EV
flightInfo.org_dest.EV <- flightInfo %>%
  filter(UniqueCarrier == 'EV') %>%
  select(FlightNum, Origin, Dest, UniqueCarrier) %>%
  distinct(FlightNum, Origin, Dest) %>%
  group_by(Origin,Dest) %>%
  summarize(cnt=n())

map("world", col="#191919", fill=TRUE, bg="#333300", lwd=0.05, xlim=xlim, ylim=yli
flightInfo.org_dest.EV<-flightInfo.org_dest.EV[order(flightInfo.org_dest.EV$cnt),]
maxcnt <- max(flightInfo.org_dest.EV$cnt)
for (j in 1:length(flightInfo.org_dest.EV$Origin)) {
  air1 <-
    airports[airports$iata == as.character(flightInfo.org_dest.EV[j,]$Origin),]
  air2 <-
    airports[airports$iata == as.character(flightInfo.org_dest.EV[j,]$Dest),]
```

```
  inter <- gcIntermediate(c(air1[1,]$long, air1[1,]$lat),
                          c(air2[1,]$long, air2[1,]$lat),
                          n = 100,
                          addStartEnd = TRUE
  )
  colindex <- round( (flightInfo.org_dest.EV[j,]$cnt / maxcnt) * length(colors))
  lines(inter, col = colors[colindex], lwd = 0.6)
}
```

We can see one prominent hub.

Similar plots were performed for all the airlines. After plotting flight pattern for all other carriers we observed that Southwest airlines (WN) operated without a hub system, hubs for other airlines are prominent, but small carriers tend to operate locally.

Determine if flight arrival and departure delay is governed by average flight distance between origin and destination airports

In the operation below, the dataset was grouped based on origin and destination airports. The next step involved in generating mean(Distance), mean(DepDelay) and mean(ArrDelay).

```
flights.08 = sample
```

```
flights_by_distance <- flights.08 %>%
  group_by(Origin,Dest) %>%
  summarise(no_of_flights=n(),avg_distance=mean(Distance),
            avg_departure_delay=mean(DepDelay),
            avg_arrival_delay=mean(ArrDelay))
```

Mutating a new variable 'route' from 'Origin' and 'Destination'. Arranging the results based on 'average_distance'.

```
flights_by_distance_arranged <- flights_by_distance %>% mutate(route=paste(as.char
  top_n(n=5,wt=avg_distance) %>%
  arrange(desc(avg_distance))
```

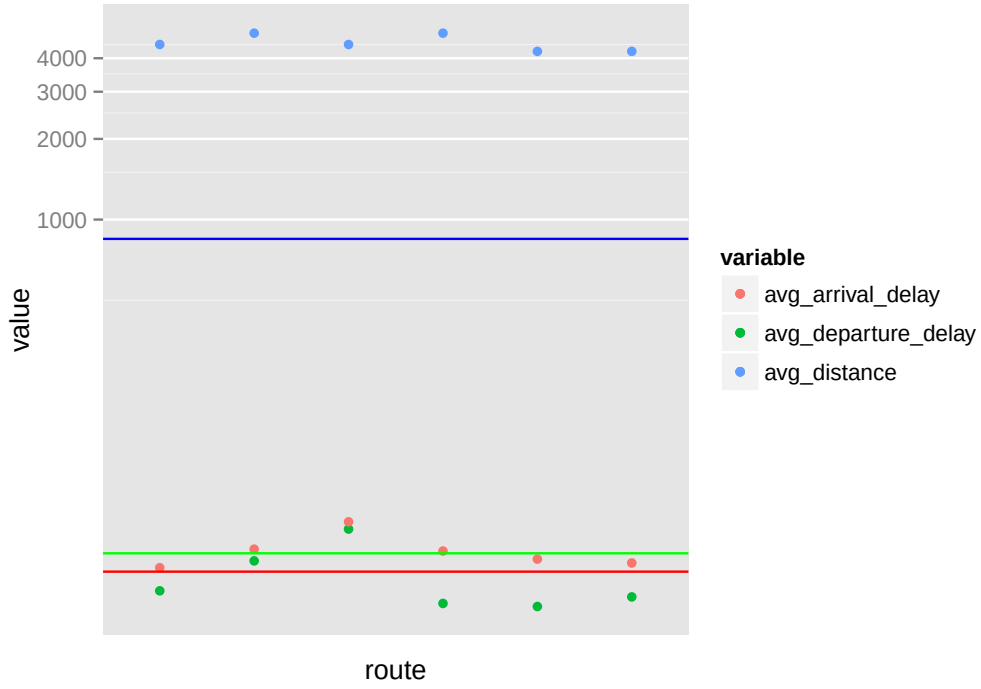
Filtering the resultset for better visualization.

```
filtered_long_distance_flight_results <- flights_by_distance_arranged   %>%
  filter(avg_distance>2700)
```

Importing 'ggplot2' library for visualization

```
library(ggplot2)
```

```
ggplot(filtered_long_distance_flight_results,aes(route,y=value,color=variable))+
  geom_point(aes(y=avg_distance,col="avg_distance"))+
  geom_point(aes(y=avg_departure_delay,col="avg_departure_delay")) +
  geom_point(aes(y=avg_arrival_delay,col="avg_arrival_delay")) +
  geom_abline(aes(color="mean (departure delay)"),intercept = 48.53711, slope = 0,
  geom_abline(aes(color="mean (arrival delay)"),intercept = 56.82167, slope = 0,co
  geom_abline(aes(color="mean (average distance)"),intercept = 847.0752, slope = 0
  coord_trans(y="log2")+
  scale_x_discrete(breaks=NULL)
```



The goal here is to identify trends that may exist for delays suffered by flights in correspondence to distance. The plot represents that the 'average arrival delay' is equal to or more than the 'mean (average arrival delay)' for most of the higher values of 'average distance' between origin and destination airports.

Determine if flight arrival and departure delay increases with an increase in the number of flights

Grouping the dataset based on Month and DayofMonth and generating the mean arrival and departure delays for all days of the year.

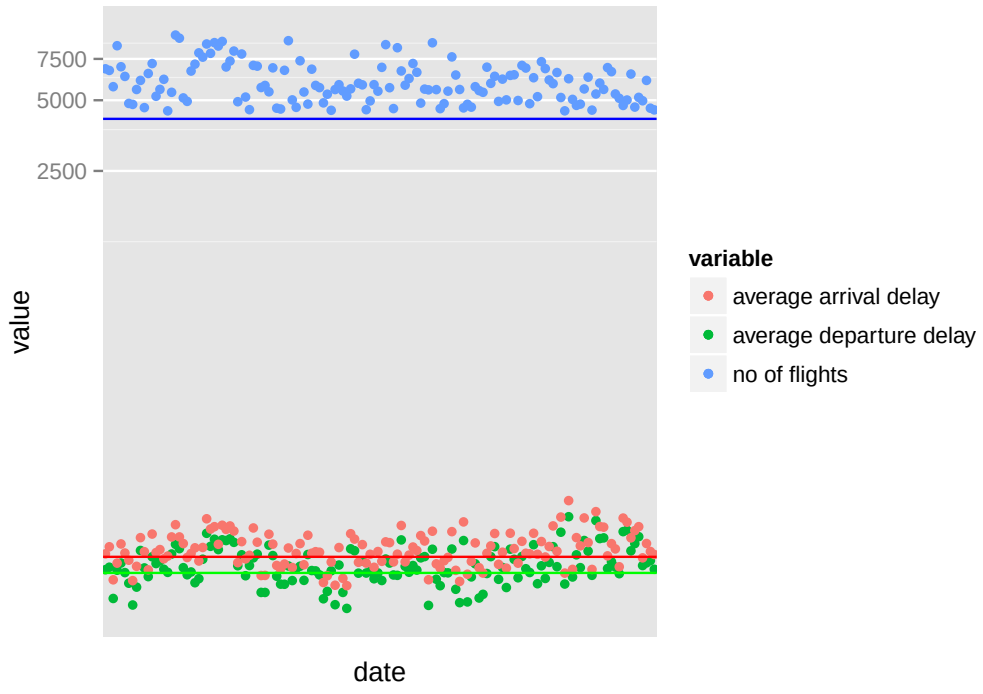
```
no_flights_in_a_day <- flights.08 %>%
  group_by(Month, DayofMonth) %>%
  summarise(no_flights = n(),avg_departure_delay = mean(DepDelay),avg_arrival_dela
  mutate(date=paste(as.character(Month),",",as.character(DayofMonth))) %>%
  arrange(desc(no_flights))
```

Limiting the available results to fewer values for better visualization purposes.

```
filtered_flight_results <- no_flights_in_a_day %>%
  filter(no_flights > 4500)
```

Visualizing the average arrival and departure delays compared to the number of flights. The plot also displays the mean values of average arrival delay, average departure delay and no of flights.

```
ggplot(filtered_flight_results,aes(date,y=value,color=variable))+
  geom_point(aes(y=no_flights,col="no of flights"))+
  geom_point(aes(y=avg_departure_delay,col="average departure delay")) +
  geom_point(aes(y=avg_arrival_delay,col="average arrival delay")) +
  geom_abline(aes(color="mean (average departure delay)"),intercept = 48.53711, sl
  geom_abline(aes(color="mean (average arrival delay)"),intercept = 56.82167, slop
  geom_abline(aes(color="mean (average no of flights)"),intercept = 4165.943, slop
  coord_trans(y="log2")+
  scale_x_discrete(breaks=NULL)
```



For higher values of 'No of flights', the 'Average arrival delay' seems to be equal to or greater than the 'mean (average arrival delay)'.

No such conclusion can be deduced regarding 'average departure delay' as the value fluctuates with most of its values being below the 'mean (average departure delay)'.

Investigative Analysis

Investigation of Flight Cancellation using Association Mining

From the dataset description, we can see that the variable `Cancelled` is categorical in nature with binary values `0` and `1` for `Not Cancelled` and `Cancelled` respectively. We can also make a similar observation for the variable `Diverted`.

As a traveller who has some understanding of data mining techniques, we are interested to make an informed choice for avoiding travel plans that would get cancelled because of our flights being cancelled. We attempt to answer questions like *what would be the worst day and time to travel, such that the flight would be cancelled?* We make use of *Association Mining* to derive rules between factors like `Month`, `DayOfWeek`,`CRSDepTime`,`CRSArrTime`, `Origin`,`Dest`,`Distance` and various `delays` for flights operational in 2008.

Filtering the dataset for variables discussed above,

```
library(dplyr)
library(magrittr)
sample <- fread("2008.csv")
```

```
## Warning in fread("2008.csv"): Bumped column 23 to type character on data
## row 179, field contains 'A'. Coercing previously read values in this
## column from logical, integer or numeric back to character which may not
## be lossless; e.g., if '00' and '000' occurred before they will now be just
## '0', and there may be inconsistencies with treatment of ',,' and ',NA,' too
## (if they occurred in this column before the bump). If this matters please
## rerun and set 'colClasses' to 'character' for this column. Please note
## that column type detection uses the first 5 rows, the middle 5 rows and the
## last 5 rows, so hopefully this message should be very rare. If reporting to
## datatable-help, please rerun and include the output from verbose=TRUE.
```

```
##
Read 10.8% of 7009728 rows
Read 29.5% of 7009728 rows
Read 48.4% of 7009728 rows
Read 67.0% of 7009728 rows
Read 85.7% of 7009728 rows
Read 7009728 rows and 29 (of 29) columns from 0.642 GB file in 00:00:07
```

```
sample <- sample %>%
  select(
    Month,DayOfWeek,CRSDepTime,
    CRSArrTime,ArrDelay,DepDelay,
    Origin,Dest,Distance,Cancelled,Diverted,
    CarrierDelay,WeatherDelay,NASDelay,SecurityDelay,LateAircraftDelay
  )
```

Eventhough the variables in consideration appear categorical in nature, some the variables are treated as numerical or character. We clean the sampled dataset to convert the variables in categorical variables. We made use of `quantile()` function in r to determine intervals for numerical variables, values falling between the calculated intervals were treated as a category.

```
sample$Month <- as.factor(sample$Month)

#Converting DayOfWeek into category
sample$DayOfWeek <- as.factor(sample$DayOfWeek)

#CRSDepTime to categorical
quantile(sample$CRSDepTime)
```

```
##   0%  25%  50%  75% 100%
##    0  925 1320 1715 2359
```

```
sample$CRSDepTime <- ordered(cut
                             (
                             sample$CRSDepTime,
                             c(-1, 600, 1200, 1800, 2400),
                             labels = c("Overnight","Morning",
```

```
                                        "Afternoon","Evening")
                                ))
sample$CRSDepTime <- as.factor(sample$CRSDepTime)

#CRSArrTime to categorical
quantile(sample$CRSArrTime)
```

```
##   0%  25%  50%  75% 100%
##    0 1115 1517 1907 2400
```

```
sample$CRSArrTime <- ordered(cut
                                (
                                sample$CRSArrTime,
                                c(-1, 600, 1200, 1800, 2400),
                                labels = c("Overnight","Morning",
                                           "Afternoon","Evening")
                                ))
sample$CRSArrTime <- as.factor(sample$CRSArrTime)


#Converting into categories for Distance

quantile(sample$Distance)
```

```
##   0%  25%  50%  75% 100%
##   11  325  581  954 4962
```

```
sample$Distance <- ordered(cut(sample$Distance,
                                  c(0,300,600,1000, Inf)),
                           labels = c("Short","Medium", "Long", "Too Long"))
sample$Distance <- factor(sample$Distance)

#Converting into categories for ArrDelay
quantile(sample$ArrDelay, na.rm = T)
```

```
##   0%  25%  50%  75% 100%
## -519  -10   -2   12 2461
```

```
sample$ArrDelay <- ordered(
  cut
  (sample$ArrDelay,
  c(-600,-1,0,25, 80, Inf)),
  labels = c(
    "Early","On-Time","Delayed",
    "Delayed Intermediately","Huge Delays"
  )
)
sample$ArrDelay <- as.factor(sample$ArrDelay)

#Converting DepDelay into categories
quantile(sample$DepDelay, na.rm = T)
```

```
##   0%  25%  50%  75% 100%
## -534   -4   -1    8 2467
```

```
sample$DepDelay <- ordered(
  cut
  (sample$DepDelay,
  c(-600,-1,0,25, 80, Inf)),
  labels = c(
    "Early","On-Time","Delayed",
    "Delayed Intermediately","Huge Delays"
  )
)
sample$DepDelay <- as.factor(sample$DepDelay)

#Converting Carrier Delay into category
quantile(sample$CarrierDelay, na.rm = T)
```

```
##   0%  25%  50%  75% 100%
##    0    0    0   16 2436
```

```
sample$CarrierDelay <- ordered(cut
                               (sample$CarrierDelay,
                               c(-100,0,16,Inf)),
                               labels = c("No-Delay","Delay","Huge-Delay"))
```

```
sample$CarrierDelay <- as.factor(sample$CarrierDelay)

#Converting Weather Delay into category
quantile(sample$WeatherDelay, na.rm = T)
```

```
##   0%  25%  50%  75% 100%
##    0    0    0    0 1352
```

```
sample$WeatherDelay <- ordered(cut
                               (sample$WeatherDelay,
                               c(-100,0,16,Inf)),
                               labels = c("No-Delay","Delay","Huge-Delay"))
sample$WeatherDelay <- as.factor(sample$WeatherDelay)

#Converting NASDelay into category
quantile(sample$NASDelay, na.rm = T)
```

```
##   0%  25%  50%  75% 100%
##    0    0    6   21 1357
```

```
sample$NASDelay <- ordered(cut
                           (sample$NASDelay,
                           c(-100,0,21,Inf)),
                           labels = c("No-Delay","Delay","Huge-Delay"))
sample$NASDelay <- as.factor(sample$NASDelay)

#Security Delay to Category
quantile(sample$SecurityDelay, na.rm = T)
```

```
##   0%  25%  50%  75% 100%
##    0    0    0    0  392
```

```
sample$SecurityDelay <- ordered(cut
                                (sample$SecurityDelay,
                                c(-100,0,21,Inf)),
                                labels = c("No-Delay","Delay","Huge-Delay"))
sample$SecurityDelay <- factor(sample$SecurityDelay)
```

```
#Converting LateAircraftDelay into category
quantile(sample$LateAircraftDelay, na.rm = T)
```

```
##   0%  25%  50%  75% 100%
##    0    0    0   26 1316
```

```
sample$LateAircraftDelay <- ordered(cut
                                    (sample$LateAircraftDelay,
                                    c(-100,0,26,Inf)),
                                    labels = c("No-Delay","Delay","Huge-Delay"))
sample$LateAircraftDelay <- as.factor(sample$LateAircraftDelay)

#Origin
sample$Origin <- as.factor(sample$Origin)

#Dest
sample$Dest <- as.factor(sample$Dest)

#Cancelled
sample$Cancelled <- as.factor(sample$Cancelled)

#Diverted
sample$Diverted <- as.factor(sample$Diverted)

#Org
sample$Origin <- as.factor(sample$Origin)

#Dest
sample$Dest <- as.factor(sample$Dest)
```

We also observed that for all records where the status of the flights was `Cancelled==1`, variables `CarrierDelay`,`WeatherDelay`,`NASDelay`,`SecurityDelay`,`LateAircraftDel` correspond to `NA` values. Hence, we decided to try and find relations between variables `Month`,`DayOfWeek`,`CRSArrTime`,`CRSDepTime` for flight cancellations by creating a new sample as shown below.

```
sample.association <- sample %>%
  select(Month,DayOfWeek,CRSArrTime,CRSDepTime,Cancelled)
```

We apply `apriori` algorithm to generate rules on `sample.association` dataset. Since the ratio of records for flights that are cancelled is very very less as compared to total number of records, we consider very low values of parameters support and confidence. We are also interested in rules that lead to cancellation of flight, hence assign the `rhs` as `Cancelled=1`.

```
library(arules)
```

```
## Loading required package: Matrix
##
## Attaching package: 'arules'
##
## The following objects are masked from 'package:base':
##
##     %in%, write
```

```
apriori.appearance = list(rhs = c('Cancelled=1'), default = 'lhs')
apriori.parameter = list(
  support = 0.000001,confidence = 0.00000001,minlen = 1,maxlen = 5
)
rules = apriori(sample.association,parameter = apriori.parameter,appearance =
                  apriori.appearance)
```

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##       1e-08    0.1    1 none FALSE            TRUE   1e-06      1      5
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## apriori - find association rules with the apriori algorithm
```

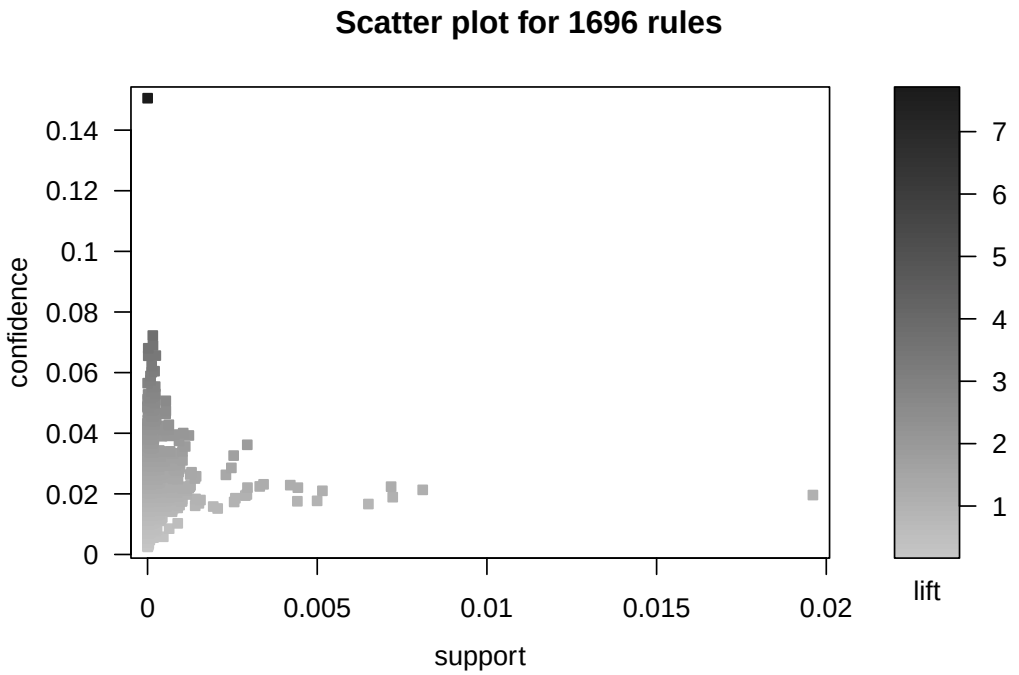
```
## version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[29 item(s), 7009728 transaction(s)] done [1.11s].
## sorting and recoding items ... [29 item(s)] done [0.14s].
## creating transaction tree ... done [4.74s].
## checking subsets of size 1 2 3 4 5 done [0.00s].
## writing ... [1696 rule(s)] done [0.00s].
## creating S4 object  ... done [0.45s].
```

Lets see the scatter plot for the rules generated,

```
library(arulesViz)
```

```
## Loading required package: grid
##
## Attaching package: 'arulesViz'
##
## The following object is masked from 'package:base':
##
##     abbreviate
```

```
plot(rules)
```



We can observe from the above plot, most rules are for lower values of support in the range `0` to `0.02`. This is because the dataset consists of higher concentration of records with status of cancellation as `0`.

Lets see some top rules based on *Confidence*,

```
inspect(sort(rules, by = 'confidence')[1:5])
```

```
##   lhs                          rhs                support confidence     lift
## 1 {Month=12,
##    DayOfWeek=3,
##    CRSArrTime=Overnight,
##    CRSDepTime=Afternoon} => {Cancelled=1} 1.997224e-06 0.15053763 7.678070
## 2 {Month=2,
##    DayOfWeek=2,
##    CRSArrTime=Evening,
##    CRSDepTime=Evening}   => {Cancelled=1} 1.542143e-04 0.07223522 3.684308
## 3 {Month=2,
##    DayOfWeek=2,
##    CRSDepTime=Evening}   => {Cancelled=1} 1.597780e-04 0.06895702 3.517106
```

```
## 4 {Month=2,
##    DayOfWeek=3,
##    CRSArrTime=Morning,
##    CRSDepTime=Overnight} => {Cancelled=1} 2.040022e-05 0.06803045 3.469847
## 5 {Month=2,
##    DayOfWeek=2,
##    CRSArrTime=Evening}   => {Cancelled=1} 2.432334e-04 0.06560973 3.346380
```

From the above result, considering relatively higher value for confidence, I would avoid flight bookings in the Month of **December** for flights that take of during **Afternoon**. We also observe a significant large value for *lift* which means flights are getting cancelled for the mentioned Month and departure interval with a frequency higher than expected. Generally, we can also observe that **February** also records significant cancellations for flights departing during evenings.

Decision Tree

Decision trees have a number of advantages. They are what's known as a glass-box model, after the model has found the patterns in the data you can see exactly what decisions will be made for unseen data that you want to predict. They are also intuitive and can be read by people with little experience in machine learning after a brief explanation. Finally, they are the basis for some of the most powerful and popular machine learning algorithms. They can be used for classification (categorical variables) or regression (continuous variables) applications.

The following code walks through how to setup and analyze a decision tree that has been created using the flights dataset from the year of 2008 . Applying decision tree algorithms to our dataset will help us to understand more about flights on-time performance and the factors that contribute to flight delays.

We use rpart for 'Recursive Partitioning and Regression Trees' and uses the CART decision tree algorithm. While rpart comes with base R, you still need to import the functionality each time you want to use it.

loading these packages into R:

```
library(rpart)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.1

## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(rpart.plot)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.2.1

## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(maptree)
```

```
## Warning: package 'maptree' was built under R version 3.2.1

## Loading required package: cluster
```

```
library(party)
```

```
## Warning: package 'party' was built under R version 3.2.1

## Loading required package: mvtnorm
## Loading required package: modeltools
## Loading required package: stats4
##
## Attaching package: 'modeltools'
##
```

```
## The following object is masked from 'package:arules':
##
##     info
##
## Loading required package: strucchange

## Warning: package 'strucchange' was built under R version 3.2.1

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 3.2.1

##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Loading required package: sandwich

## Warning: package 'sandwich' was built under R version 3.2.1

library(partykit)

## Warning: package 'partykit' was built under R version 3.2.1

##
## Attaching package: 'partykit'
##
## The following objects are masked from 'package:party':
##
##     cforest, ctree, ctree_control, edge_simple, mob, mob_control,
##     node_barplot, node_bivplot, node_boxplot, node_inner,
##     node_surv, node_terminal
```

```
library(Formula)
```

Reading the whole file for the purpose of our analysis:

```
data = fread("2008.csv")
```

```
## Warning in fread("2008.csv"): Bumped column 23 to type character on data
## row 179, field contains 'A'. Coercing previously read values in this
## column from logical, integer or numeric back to character which may not
## be lossless; e.g., if '00' and '000' occurred before they will now be just
## '0', and there may be inconsistencies with treatment of ',,' and ',NA,' too
## (if they occurred in this column before the bump). If this matters please
## rerun and set 'colClasses' to 'character' for this column. Please note
## that column type detection uses the first 5 rows, the middle 5 rows and the
## last 5 rows, so hopefully this message should be very rare. If reporting to
## datatable-help, please rerun and include the output from verbose=TRUE.
```

```
##
Read 13.3% of 7009728 rows
Read 33.4% of 7009728 rows
Read 53.4% of 7009728 rows
Read 73.2% of 7009728 rows
Read 93.3% of 7009728 rows
Read 7009728 rows and 29 (of 29) columns from 0.642 GB file in 00:00:07
```

```
data = na.omit(data)
```

Since we need categorical data for analysis, we have subdivided the interesting variables into ranges for indepth analysis:

The categorical values for specific range of values for multiple variable was decided based on a cursory view of the data set by using the 'View()' function.

The 'Distance' variable is divided into subgroups of certain numeric divisions.

```
data$Distance= ordered(cut(data$Distance, c(0,500,1000, Inf)),
                       labels = c("upto500", "500to1000","1000 and more"))
```

We replace numeric values that are actually categorical in nature by using 'gsub()'. The function 'gsub()' performs replacement of the first and all matches respectively.

```
data$DayOfWeek = gsub("1", "Sunday", data$DayOfWeek)

data$DayOfWeek = gsub("2", "Monday", data$DayOfWeek)


data$DayOfWeek = gsub("3", "Tuesday", data$DayOfWeek)


data$DayOfWeek = gsub("4", "Wednesday", data$DayOfWeek)


data$DayOfWeek = gsub("5", "Thursday", data$DayOfWeek)

data$DayOfWeek = gsub("6", "Friday", data$DayOfWeek)

data$DayOfWeek = gsub("7", "Saturday", data$DayOfWeek)
```

The 'Origin' Origin airports are classified into region wise sub-categories. These locations are categorized in the form of "South West", "Mid west", "West", "East", "South east" regions. We use the function gsub to perform these operations:

```
data$Origin = gsub("ABQ| AMA| AUS| CRP| DAL|
                    ELP| HOU| HRL| LBB| OKC| SAT|
                    TUS| TUL| MAF| IAH| DFW| BRO|
                    CHS| TYS", "SouthWest", data$Origin)

data$Origin = gsub("BDL| BUF| ISP| MHT| PHL|
                    PIT| PVD| ALB| ROC| EWR| BTV|
                    BGR| SYR| BOS| ABE| PWM| LGA|
                    JFK| MDT", "NorthEast", data$Origin)

data$Origin = gsub("BHM| BNA| BWI| FLL| IAD|
                    JAN| JAX| LIT| MCO| RDU| TPA|
                    ORF| PBI| SDF| RSW| ATL| RIC|
```

```
                    MIA| CLT", "SouthEast", data$Origin)

data$Origin = gsub("CLE| CMH| DTW| IND| MCI|
                    MDW| STL| OMA| MKE| DAY| DSM|
                    GRR| ORD| MSP| MSN| ICT| ATW|
                    CAK| CID", "MiddleWest", data$Origin)


data$Origin = gsub("BOI| BUR| DEN| LAS| LAX| MSY|
                    OAK| ONT| PDX| RNO| SAN|
                    SEA| SFO| SJC| SLC| SMF| SNA|
                    GEG| LFT", "West", data$Origin)
```

We perform the same operation to categorize the 'Dest' Destination of airports. These airports are categorized into "South West", "Mid west", "West", "East", "South east" regions. We use the function gsub to perform these operations:

```
data$Dest = gsub("ABQ| AMA| AUS| CRP| DAL| ELP| HOU| HRL| LBB| OKC| SAT|
                  TUS| TUL| MAF| IAH| DFW| BRO| CHS| TYS", "SouthWest", data$Dest

data$Dest = gsub("BDL| BUF| ISP| MHT| PHL| PIT| PVD| ALB| ROC| EWR| BTV|
                  BGR| SYR| BOS| ABE| PWM| LGA| JFK| MDT", "NorthEast", data$Dest

data$Dest = gsub("BHM| BNA| BWI| FLL| IAD| JAN| JAX| LIT| MCO| RDU| TPA|
                  ORF| PBI| SDF| RSW| ATL| RIC| MIA| CLT", "SouthEast", data$Dest

data$Dest = gsub("CLE| CMH| DTW| IND| MCI| MDW| STL| OMA| MKE| DAY| DSM|
                  GRR| ORD| MSP| MSN| ICT| ATW| CAK| CID", "MiddleWest", data$Des


data$Dest = gsub("BOI| BUR| DEN| LAS| LAX| MSY| OAK| ONT| PDX| RNO| SAN|
                  SEA| SFO| SJC| SLC| SMF| SNA| GEG| LFT", "West", data$Dest)
```

The variable 'DepDelay' is divided into low and high Delay sub-categories.

```
data$DepDelay = ordered(cut(data$DepDelay, c(-Inf, 60, Inf)),
                        labels = c("low","high"))
```

Converting other available variables as factor variables

```
data$DayofMonth = as.factor(data$DayofMonth)
```

Rpart

The rpart tree uses recursive partitioning to identify the best nodes to create-within the tree. When attempting to run therparttree with the completedataset, only one node, the root node is found.

A formula based on our dataset may be created.

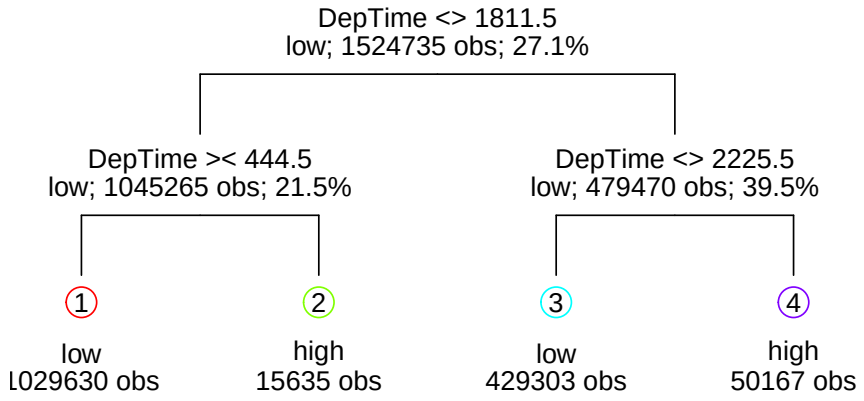
```
formula3 = DepDelay ~ DayofMonth + DayOfWeek + DepTime + Distance
```

We will run the rpart function and analyze and plot the result.

```
rpartcalculation = rpart(data = data, formula= formula3)
```

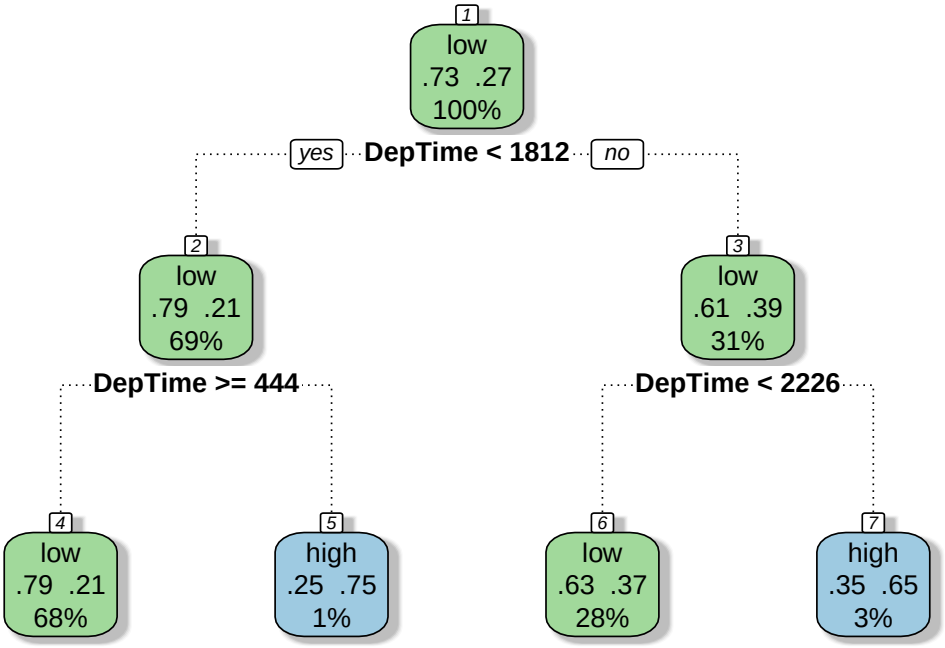
Drawing the tree

```
draw.tree(rpartcalculation, nodeinfo = TRUE)
```



Decision Tree Fancy Plot

```
fancyRpartPlot(rpartcalculation)
```



Printing the results of our calculation:

```
print(rpartcalculation)
```

```
## n= 1524735
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 1524735 413541 low (0.7287784 0.2712216)
##   2) DepTime< 1811.5 1045265 224229 low (0.7854812 0.2145188)
##     4) DepTime>=444.5 1029630 212484 low (0.7936307 0.2063693) *
##     5) DepTime< 444.5 15635   3890 high (0.2488008 0.7511992) *
##   3) DepTime>=1811.5 479470 189312 low (0.6051640 0.3948360)
##     6) DepTime< 2225.5 429303 156804 low (0.6347475 0.3652525) *
##     7) DepTime>=2225.5 50167  17659 high (0.3520043 0.6479957) *
```

From this analysis we can conclude that, normally at night after 10.30pm delays are generally more as compared to the flights delays that occur during daytime. This can hint that the delays could be more inclined to the arrival time and departure times for a aircraft, than the actual air time for a particular flight.

```
formula4 = ArrDelay ~ DayofMonth + DayOfWeek + Distance + Origin
```

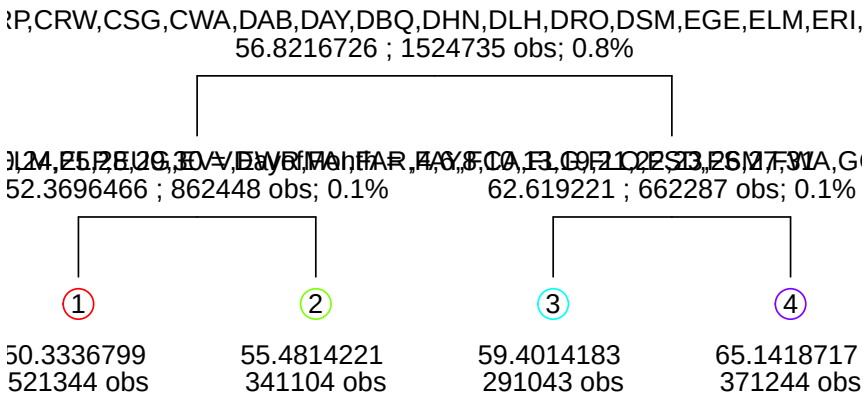
```
control4 = rpart.control(cp= 0.001)
```

Applying the rpart function on the formula and the control parameters created above.

```
rpartcalculation1= rpart(data = data, formula = formula4,
                         control = control4)
```

Displaying the decision tree:

```
draw.tree(rpartcalculation1, nodeinfo = TRUE)
```



The decision tree output is representative of the fact that our dataset is divided into specific sub parts for 'Origin' and 'DayOfWeek' variables.

Printing the tree rules:

```
print(rpartcalculation1)
```

```
## n= 1524735
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
## 1) root 1524735 4982324000 56.82167
##   2) Origin=ACT,ADK,ADQ,AKN,ANC,ATL,AUS,BET,BFL,BIL,BIS,BLI,BNA,BRW,BTM,BUR,BWI
##     4) DayofMonth=1,2,3,5,7,9,11,12,14,15,16,17,18,20,24,25,28,29,30 521344 129
##     5) DayofMonth=4,6,8,10,13,19,21,22,23,26,27,31 341104 1061374000 55.48142 *
##   3) Origin=ABE,ABI,ABY,ACK,ACV,ACY,AEX,AGS,ALB,ALO,AMA,ASE,ATW,AVL,AVP,AZO,BGM
##     6) Origin=ABY,ALB,AMA,BOS,BQK,BQN,BZN,CDC,CHS,CMH,CRP,CSG,DAY,DLH,DRO,ERI,E
##     7) Origin=ABE,ABI,ACK,ACV,ACY,AEX,AGS,ALO,ASE,ATW,AVL,AVP,AZO,BGM,BGR,BJI,B
```

Ctree

Ctree is a conditional inference trees which embed tree- structured regression models into a well defined theory of conditional inference procedure.

We decided to use the ctree command to look at a decision tree. The max depth argument was set to three, indicating the maximum number of decision levels from the root node to a terminal node is three.

We chose three simply because the decision tree plot becomes crowded with a greater depth. A depth of three still allows for insights into the data without the user experiencing extreme cognitive load due to the clutter.

Departure Delay:

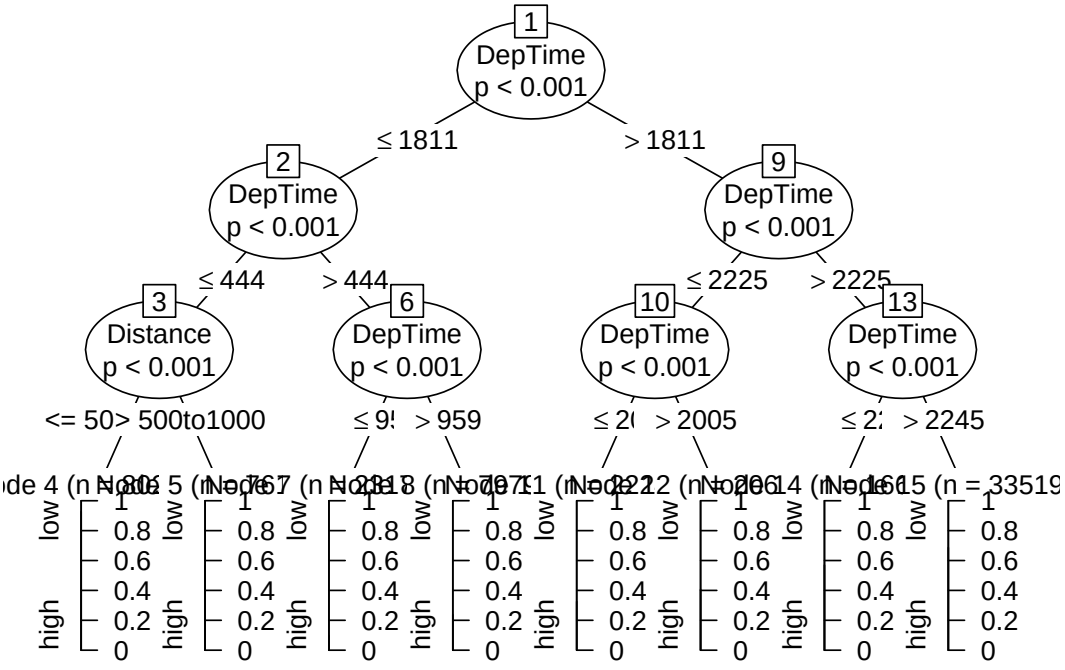
```
formula1 = DepDelay ~ Distance + DepTime
control1 = ctree_control(maxdepth = 3)
```

Building the ctree:

```
ctree1 = ctree(data = data, formula = formula1, control = control1)
```

Plotting the created ctree:

```
plot(ctree1)
```



As explained previously using rpart functionlities, we can visualize that delays are more frequent in the night time than delays that occur during the day time.

Arrival Delay:

Creating the formula and the control parameters:

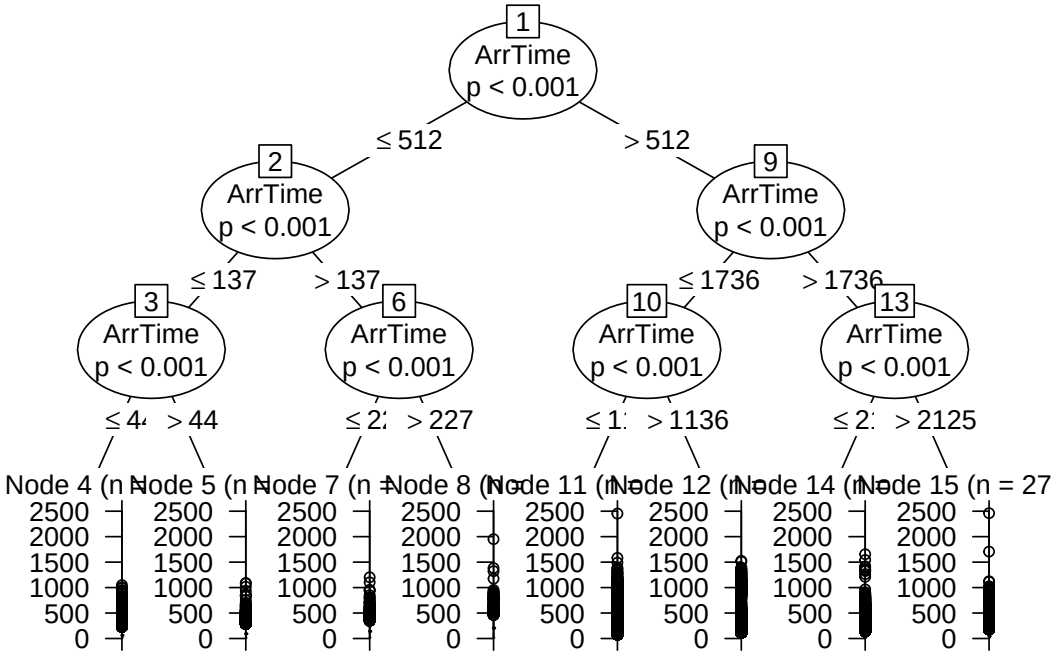
```
formula2 = ArrDelay ~ Distance + ArrTime
control2 = ctree_control(maxdepth = 3)
```

Building the ctree:

```
ctree2 = ctree(data = data, formula = formula2, control=control2)
```

Plotting the ctree:

```
plot(ctree2)
```



Through this snippet we can conclude that, most delays occur in the night as compared to day-time.

Support Vector Machines

Importing dataset

```
sample <- fread("2008.csv")
```

```
## Warning in fread("2008.csv"): Bumped column 23 to type character on data
## row 179, field contains 'A'. Coercing previously read values in this
## column from logical, integer or numeric back to character which may not
```

```
## be lossless; e.g., if '00' and '000' occurred before they will now be just
## '0', and there may be inconsistencies with treatment of ',,' and ',NA,' too
## (if they occurred in this column before the bump). If this matters please
## rerun and set 'colClasses' to 'character' for this column. Please note
## that column type detection uses the first 5 rows, the middle 5 rows and the
## last 5 rows, so hopefully this message should be very rare. If reporting to
## datatable-help, please rerun and include the output from verbose=TRUE.


##
Read 8.6% of 7009728 rows
Read 27.5% of 7009728 rows
Read 46.5% of 7009728 rows
Read 64.5% of 7009728 rows
Read 82.7% of 7009728 rows
Read 7009728 rows and 29 (of 29) columns from 0.642 GB file in 00:00:07
```

```
sample <- na.omit(sample)
data1 = sample
```

SVM is used to train a support vector machine. It can be used to carry out general regression and classification (of nu and epsilon-type), as well as density-estimation. A formula interface is provided.

In order to create a SVR model with R you will need the package e1071. So be sure to install it and to add the library(e1071) line at the start of your file.

Below is the code to make predictions with Support Vector Regression:

'ArrDelay' is kept as the response vairable and is further categorized into low and high.

```
data1$ArrDelay = ordered(cut(data1$ArrDelay, c(10,40, Inf)), labels= c("Low", "Hig
```

A subset of data has been chosen for analysis considering the computing power of our machines.

```
data.frame = data1[sample(nrow(data1),1000), ]
```

```
subset = nrow(data.frame)
```

Now we split the data into a training set (70%) and a test set (30%):

```
nxd.train = sample(1:subset, 0.7 * subset)
```

```
training = data.frame[nxd.train, ]
test = data.frame[-nxd.train, ]
```

The following libraries need to be used to perform SVM on the dataset:

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.2.1
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.1
```

```
## Loading required package: lattice
```

'ArrDelay' will be directly based on the distance an aircraft covers on a particular day of the week.

Hence, these are the variables that are taken into consideration.

```
formula = ArrDelay ~ DayOfWeek + Distance
```

```
plot.formula = DayOfWeek ~ Distance
```

We called the svm function. The function will automatically choose SVM if it detects that the data is categorical.

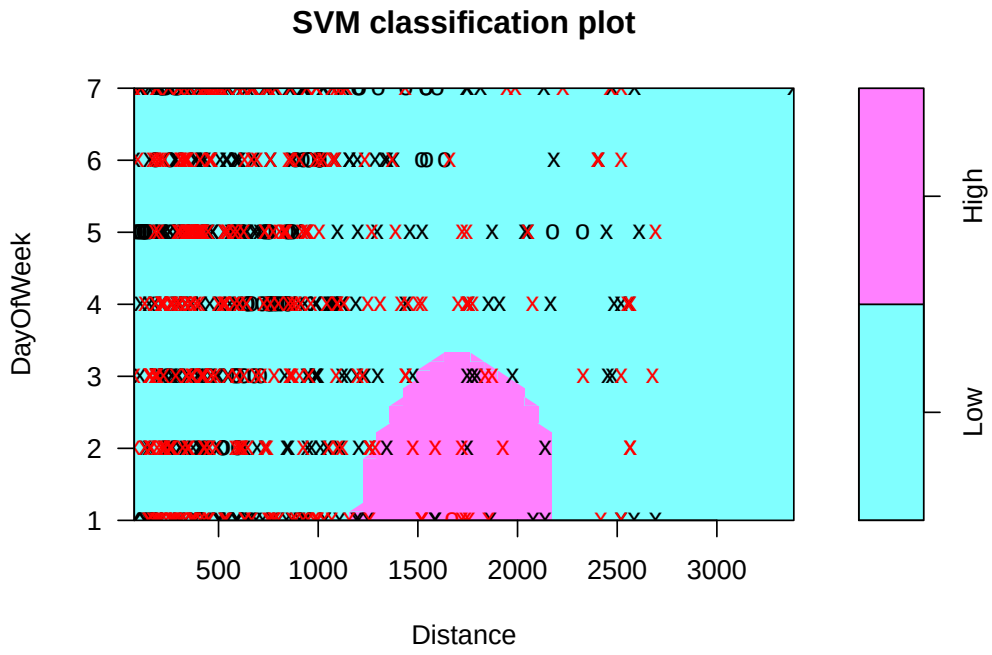
```
model = svm(formula = formula, data = training)
```

```
summary(model)
```

```
##
## Call:
## svm(formula = formula, data = training)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.5
##
## Number of Support Vectors:  632
##
##  ( 321 311 )
##
##
## Number of Classes:  2
##
## Levels:
##  Low High
```

The code draws the following graph:

```
plot(x = model, data = training, formula = plot.formula)
```



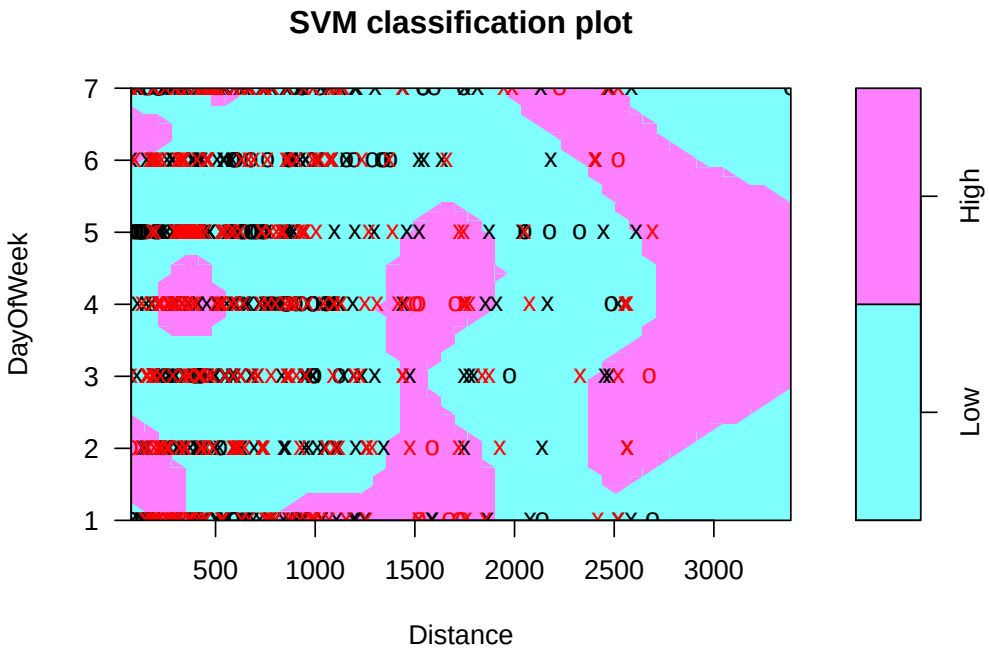
For the interpretation of a clear picture, we are changing the cost and gamma parameter.

cost is a general penalizing parameter for C-classification and gamma is the radial basis function-specific kernel parameter.

```
model1 = svm(formula = formula, data = training, method = "C-classification",
             kernel = "radial", cost = 100, gamma = 1)
```

The code draws the following graph:

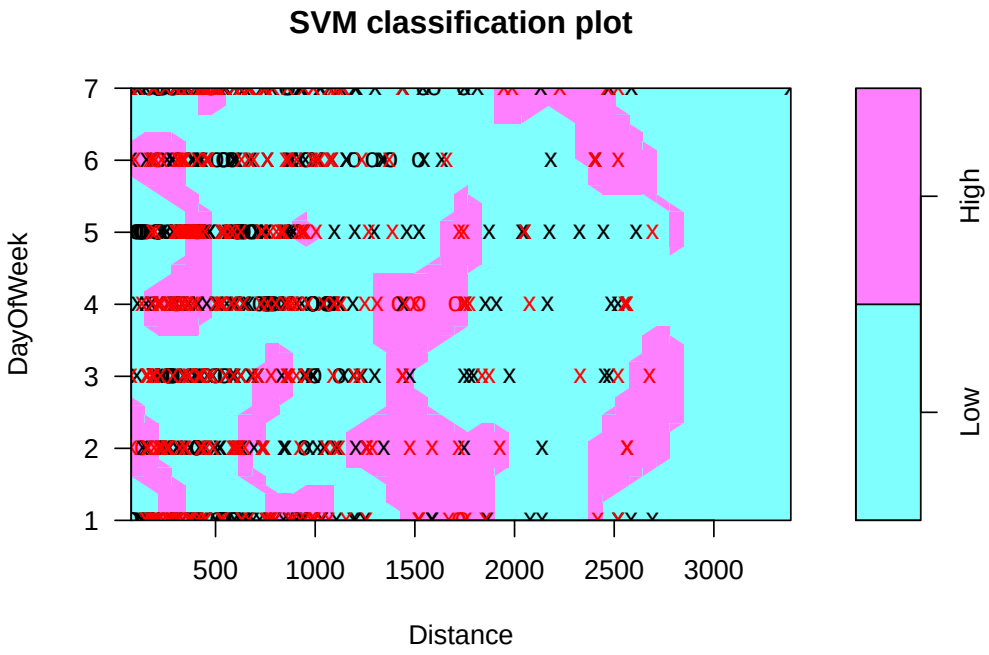
```
plot(x = model1, data = training, formula = plot.formula)
```



This plot represents that the arrival delay gets larger if the aircraft travels long distances. And the delays get higher especially on the weekends. And the delays get higher especially on the wekends. As seen from the graph, Saturday and Sunday are prone to suffer higher arrival delays than any of the other days of the week.

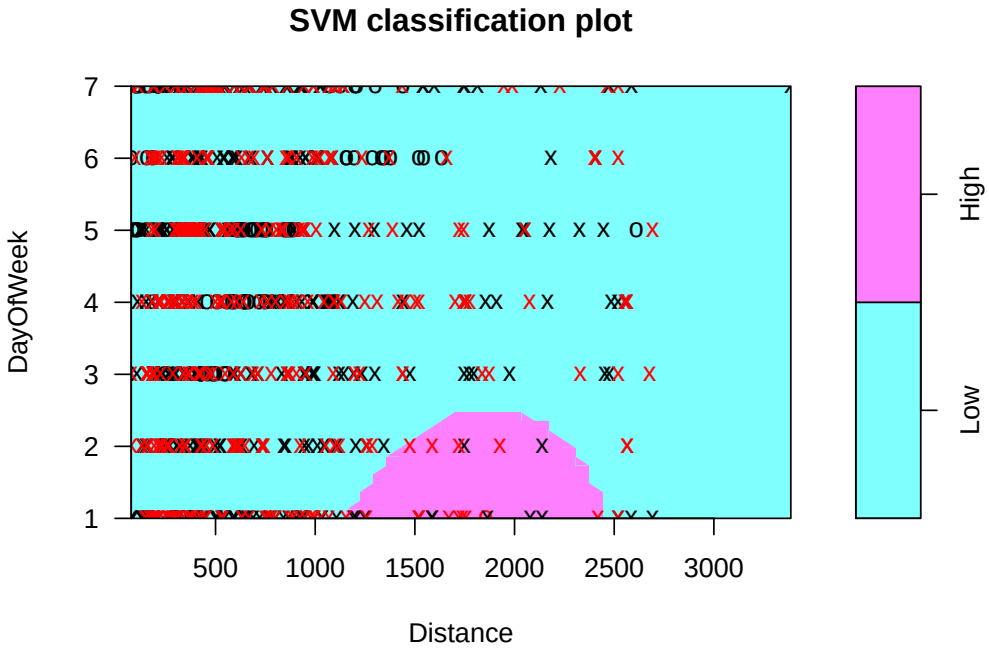
```
model2 = svm(formula = formula, data = training, method = "C-classification",
             kernel = "radial", cost = 1, gamma = 10)

plot(x = model2, data = training, formula = plot.formula)
```



This plot represents that the arrival delay gets larger if the aircraft travels long distances.

```
model3 = svm(formula = formula, data = training, method = "C-classification",
             kernel = "radial", cost = 50, gamma = 0.1)
plot(x = model3, data = training, formula = plot.formula)
```



Various plots depict the delays associated with each particular day of the week.

Summary and conclusion

Based on our analysis, we were able to derive interesting insights pertaining to arrival and departure delays associated with flights. A conclusion is presented at the end of every analysis that brings out interesting information.

Our investigation helped identify some significant factors pertaining to flight delays based on some busiest carriers as well as airports. The set of significant variables were then scrutinized further by executing drill downs into subsets of data. A lot of hidden relationships amongst the data have been tried to explore through various data mining techniques.

According to Wall Street Journal, illness, flight emergencies, and rescheduled business meetings are a big business for airline companies. At some airlines,the resulting change fee and penalties passenger ended up paying was close to $2 billion a year. If airlines can provide more information about the seasonal client data or customer's demographic information; then it would be useful to figure out a cancellation pattern, plus assist to adjust fees and penalties

based on discovered patterns. This in turn will be advantageous for generating a higher revenue based on discovered findings.