IST 687 – Introduction to Data Science

Lab Section M008 |
Group 03

SOUTHEAST CUSTOMER SERVICES ANALYSIS

Recommendations to Increase Customer Satisfaction

Submitted by: Nishit Nakrani | Prathmesh Datar | Pranjal Sondkar | Zhiwei Wang

Table of Contents

1. INTRODUCTION

Our team was given a task of helping Southeast Airlines to lower their customer churn. We were given a dataset acquired through customer survey conducted by Southeast Airlines which contained data of 10,282 flights taken by customers. The dataset had 32 variables, one of them was likelihood to recommend which shows how likely the customer is to recommend the airline to their friends. It is rated on a scale of 1 to 10, 10 being very likely and 1 being least likely. This attribute was most important as the more customers recommended this airlines to their friends, more chances the airline had to retain the existing customers and plus gaining more customers. We started with exploring the dataset to determine the assumption and business questions.

Then we cleaned the data set as many rows had NA's and also made new columns to help with further analysis. Then we ran some exploratory analysis using bar plots, text mining, etc. for which the visualizations have been included in this report. Further, we used four models to identify the leading indicators or metrics that might help keep a customer. Lastly, we used our overall analysis to meet our objective which was to provide best practices for Southeast Airlines with the main purpose of lowering customer churn.

2. ASSUMPTIONS

Not-relevant or redundant columns in the dataset:

1) **Year of First Flight:** The problem statement which was presented to us was finding who was a promoter or a detractor and for that, we as a team didn't think it is that useful to predict a promoter or a detractor.

2) **Day of Month:** In the dataset, we have a column named 'Flight.Date' which is sufficient for any analysis from the date perspective. Therefore, we didn't consider the day of the month column because we thought it is redundant.

3) **Partner.Code:** In a similar way, we thought that Partner Name offers much more information about the partner airline and therefore we didn't consider the Partner Code.

Origin City and Destination City:

We thought along with their individual impact on the overall likelihood to recommend, we need to consider their combined impact in terms of routes. Therefore, it gave us an additional quantity to look for.

Price Sensitivity:

The grade to which the price affects to customers purchasing. The price sensitivity had a range from 0 to 5. Even if we did consider it for our analysis, we didn't think this is a strong parameter to predict a promoter/detractor because sensitivity is a subjective quantity and to generalize for an entire set of audiences is difficult.

Spending:

We thought Shopping Amount and Drinking Amount at the airport should not affect the customer's opinion for the flight.

When flights get cancelled:

We assumed whenever the flight gets cancelled, the customer experiences the worst situation in terms of re-scheduled flights, longer layovers, arrival delays, and departure delays.

Passive Cases of Customers:

We didn't remove the passive cases directly for all the cases. We thought it would be beneficial if we keep passive cases as model specific.

3. BUSINESS QUESTIONS

1. Which are the top 5 and worst 5 performing cities in terms of destination and origin city?
2. Which combination of customer types makes a detractor or a promoter?
3. What are the top 5 and worst 5 routes?
4. Which is the best and the worst partner airline in terms of customer satisfaction?
5. What is the distribution of Customer satisfaction in terms of Inter and Intrastate flights?

4. DATA CLEANING & MUNGING

To maintain uniformity of data such that it supports all sorts of Descriptive Analysis and performs data modelling techniques it was important to spend some time to get the data right. We started our primitive data cleaning process by checking whether any columns have NA present in the dataset. Then we went on to create some new categorical variables which would better represent the continuous variables. Also, we created some new continuous variable columns such as ArrivalDelayRatio and DepartureDelayRatio.

4.1. Cleaning Origin City and Destination City Column

Origin City and Destination city were represented by City Name with a comma and a State Abbreviation. For map representation, we wanted just the city name and therefore removed comma and state abbreviation.

Code:

```
df$Destination.City <- gsub(",.*", "", df$Destination.City)
df$Origin.City <- gsub(",.*", "", df$Origin.City)
```

4.2. Conversion of Continuous Variables to Categorical Variables

a) <u>Type</u>: We used Likelihood.to.recommend column and made a categorical variable 'type' which included three parts. i.e Detractor, promoter and passive. We did the same by using the cut function which divides the likelihood to recommend in the range 1 – 6 as detractors, 7 – 8 as passive and 9 – 10 as promoters.

b) <u>agegroup</u>: We thought of analysing the Age column also from a bucket point of view and therefore created three age categories and stored it under the variable name 'agegroup'. Categories were Minors: for Age in between 1 – 17, Adults for Age in between 18 – 59, SrCitizen for Age 60 and above.

c) <u>Day.of.Flight:</u> We started with using the flight date column and using the 'weekdays' function to get the output in the form of Sunday, Monday, etc. and then categorized it in the form of a weekday flight or a weekend flight.

Code:

```
#  Adding type column for NPS
df <- df %>%
  mutate(type = cut(Likelihood.to.recommend,
                    breaks = c(0, 6, 8, 10),
                    labels = c("detractor","passive","promoter"))) %>%
  # 2) Adding AgeGroup column
  mutate(agegroup = cut(Age,
                        breaks = c(0, 17, 59, Inf),
                        labels = c("Minor","Adult","SrCitizen"))) %>%
  # 3) Adding Day of flight
    mutate(Day.of.Flight = weekdays(as.Date(Flight.date, '%m/%d/%Y'))) %>%
  # 4) Adding spend column for any spending at Airport
  mutate(spend = cut(Shopping.Amount.at.Airport + Eating.and.Drinking.at.Airport,
                        breaks =  c(-Inf, 0, Inf),
                        labels = c("no", "yes"))) %>%
  # 5) Updating Flight Date column in Date Format
  mutate(Flight.date = as.Date(Flight.date,"%m/%d/%y"))
# Adding Day of FLight as Weekday or Weekend
df[df$Day.of.Flight %in% c('Saturday','Sunday'),'Day.of.Flight'] <- 'weekend'
df[df$Day.of.Flight %in% c('Monday','Tuesday','Wednesday','Thursday','Friday'),
'Day.of.Flight'] <- 'weekday'
```

4.3. Removing NAs where flights were not cancelled.

There were 21 cases in the dataset where there were NAs in Arrival Delay, Departure Delay and Flight Time in minutes where the flight was not cancelled which made no sense. Therefore, we decided on removing these rows.

Code:

```
delete_list <- which(df$Flight.cancelled == 'No' & ((is.na(df$Arrival.Delay.in.Minutes)
== TRUE) | (is.na(df$Departure.Delay.in.Minutes) == TRUE) |
(is.na(df$Flight.time.in.minutes)) == TRUE))
df <- df[-delete_list,]
```

4.4. Removing NAs where flights were cancelled

Another logical presence of NAs was when flights did get cancelled and had NAs in Departure Delay, Arrival Delay and Flight time in minutes. To handle this situation, we calculated the average speed of an airplane and then we calculated Average flight time in minutes to complete the flight by dividing Flight Distance by Average Speed. Then, to replace arrival delay and departure delay, we took the maximum delay for respective columns and replaced NAs with maximum value because we believed that whenever the

flight gets cancelled, the customer experiences the worst situation in terms of re-scheduled flights, longer layovers, arrival delays, and departure delays.

Code:

```
df_test <- df %>% group_by(Origin.City, Destination.City)%>%
  summarise(n=n(),promoter_count=length(type[type == "promoter"]),
            detractor_count=length(type[type == "detractor"]),
            NPS = (promoter_count/n - detractor_count/n)*100,
            flight_distance = mean(Flight.Distance),
            flight_time = mean(Flight.time.in.minutes))
df_test <- na.omit(df_test)
df_test$flight_speed <- df_test$flight_distance/df_test$flight_time
# Average speed for a flight in miles/min
avg_flight_speed <- mean(df_test$flight_speed)

# Calculating Max delay for arrival delay and departure delay
max_dep_delay <- max(df$Departure.Delay.in.Minutes,na.rm = TRUE)
max_arr_delay <- max(df$Arrival.Delay.in.Minutes, na.rm = TRUE)
for(i in 1:nrow(df))
{
  if(df$Flight.cancelled[i] == 'Yes')
  {
    df$Flight.time.in.minutes[i] <- df$Flight.Distance[i] / avg_flight_speed
    df$Departure.Delay.in.Minutes[i] <- max_dep_delay
    df$Arrival.Delay.in.Minutes[i] <- max_arr_delay
  }
}
```

4.5. Creation of Arrival Delay Ratio and Departure Delay Ratio

Created two new variables named Arrival Delay Ratio and Departure Delay Ratio by Dividing Arrival Delay in minutes by Flight Time in Minutes. The ideology behind doing this was if the shorter flights get delayed by a few hours it will have a huge impact on customers as they will have their itinerary changed and will experience inconveniences. For longer flights, if a flight gets delayed by several minutes then it shouldn't be having the same level of impact as shorter flights. Therefore, creating these variables brought the delay on the same page.

Code:

```
df$ArrivalDelayRatio <- round(df$Arrival.Delay.in.Minutes/df$Flight.time.in.minutes,2)
df$DepartureDelayRatio<-
round(df$Departure.Delay.in.Minutes/df$Flight.time.in.minutes,2)
```

4.6. Data for Text Mining

For preparation of data for Text Mining, we only required the freeText column and type column which contained Detractor, Promoter and Passive cases

Code:

```
df <- df[,c('type','freeText')]
df <- na.omit(df)
```

5. EXPLORATORY ANALYSIS

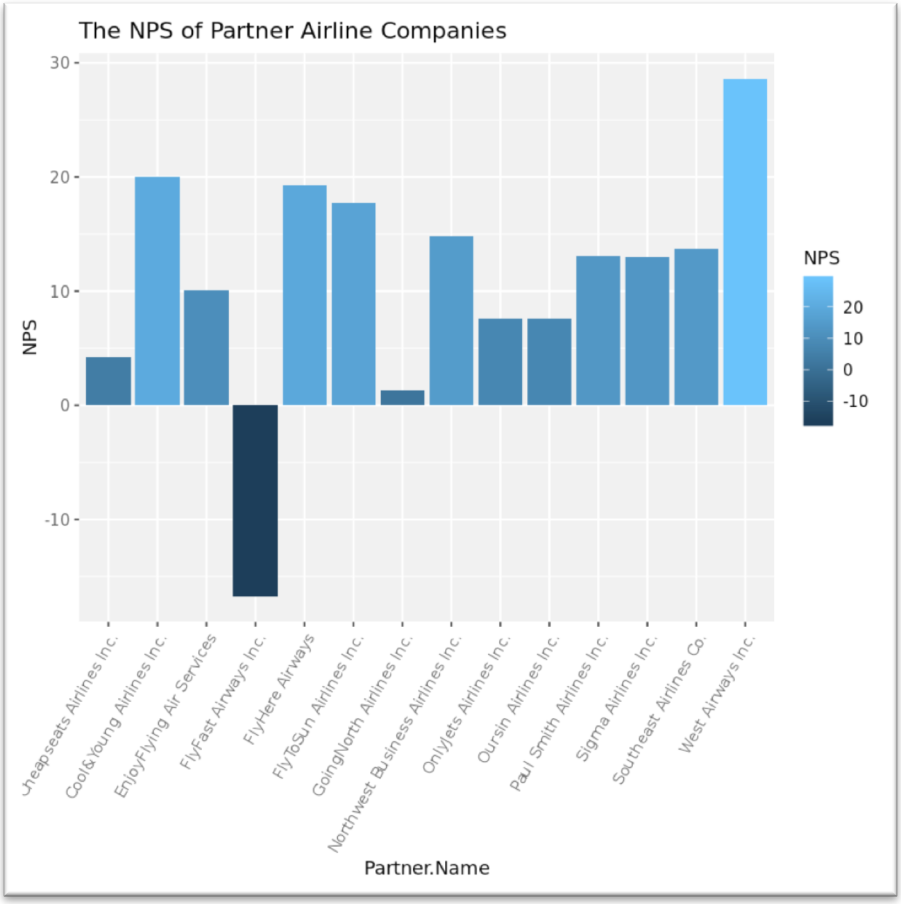
5.1. BAR CHART

1. NPS vs Single Variable Comparison

In this section, we would like to explore what kind of customers have a low Net Promoter Score (NPS). The overall NPS in a given group is computed by subtracting the percent of respondents who are detractors (score 1-7 in the likelihood to recommend) from the percent of respondents who were promoters (score 9-10). We compare the NPS of different variables in a given group through bar plots.

First, we compare and visualize the NPS of different Partner Airlines by executing the following R code:

```
partner_compare <- df %>% group_by(Partner.Name)%>%
  summarise(n=n(),promoter_count=length(type[type == "promoter"]),
            detractor_count=length(type[type == "detractor"]),
            NPS = (promoter_count/n - detractor_count/n)*100)

partner_plot <- partner_compare %>% ggplot(aes(x = Partner.Name, y = NPS,fill =
NPS)) +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  ggtitle("The NPS of Partner Airline Companies")+
  geom_col()
```

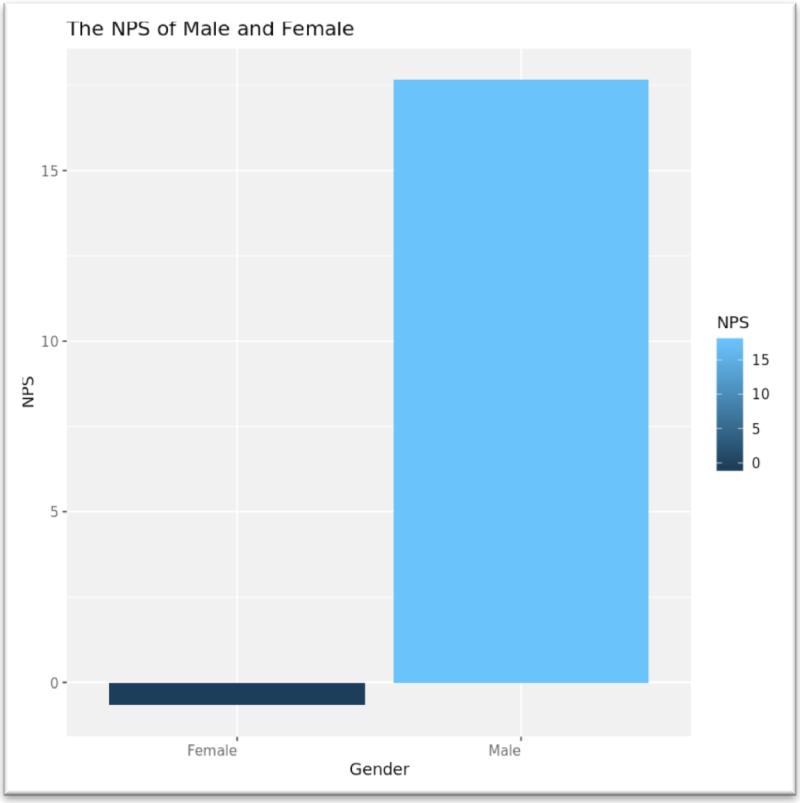


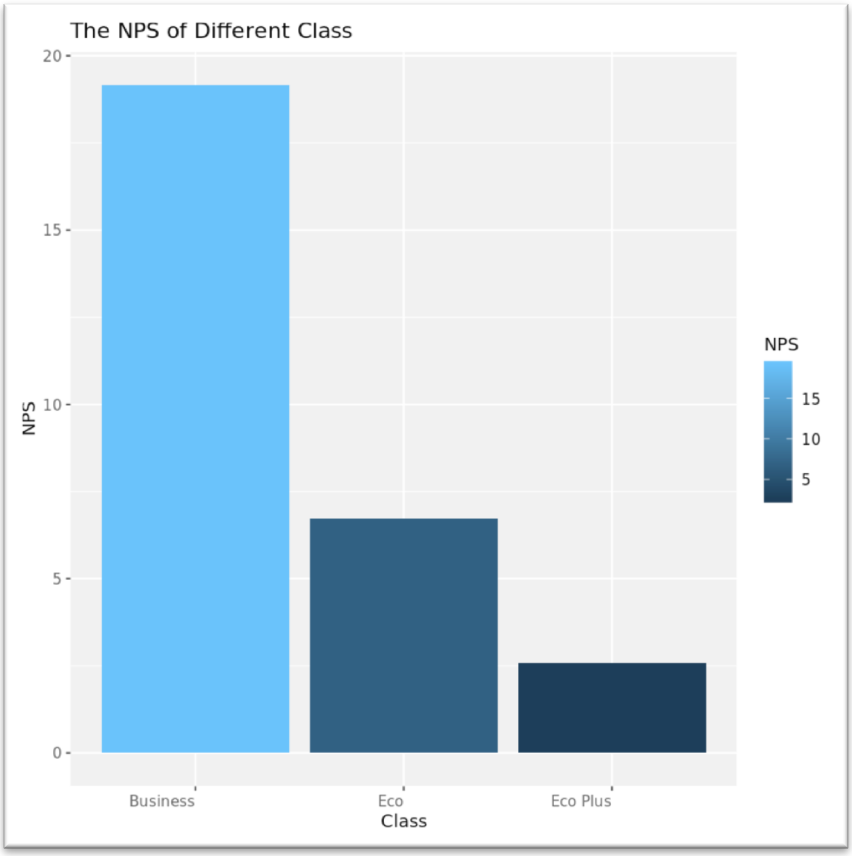
Insight: *As the graph above suggests, the 'FlyFast Airways Inc.' is the only partner airline that has a negative NPS, which means that customers travel with this particular airlines are less likely to recommend the flight to others.*

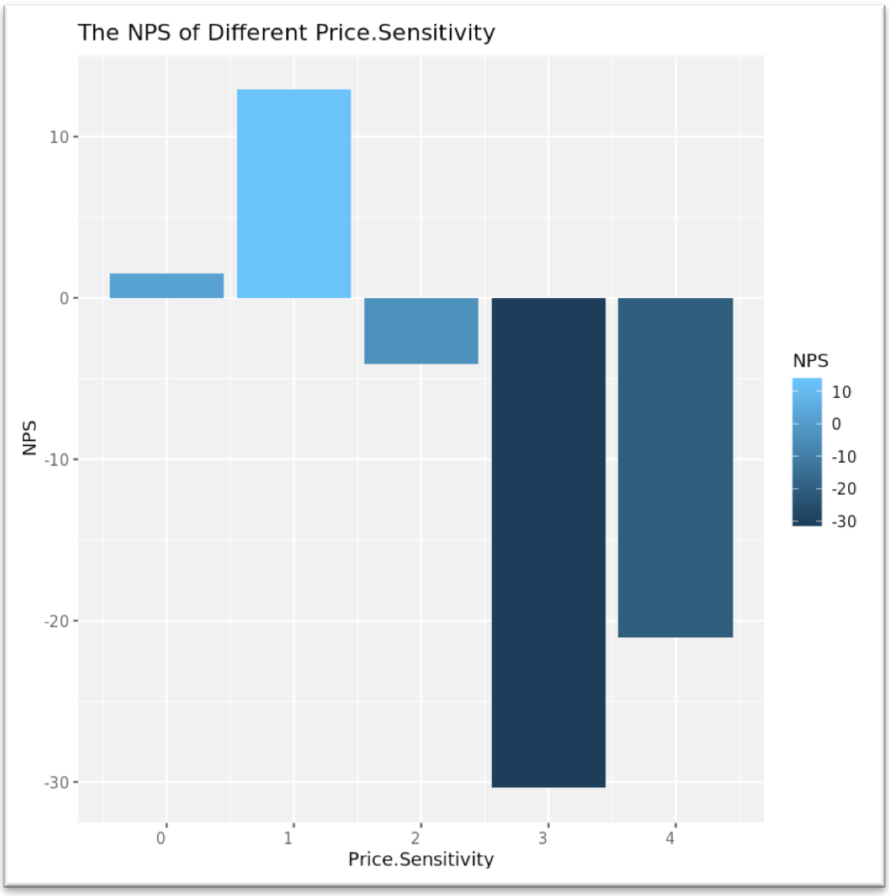
Similarly, we can compare the NPS of male and female, of different classes, of different price sensitivity, in different airline status, in three age groups, and through different types of travel by executing the following code:

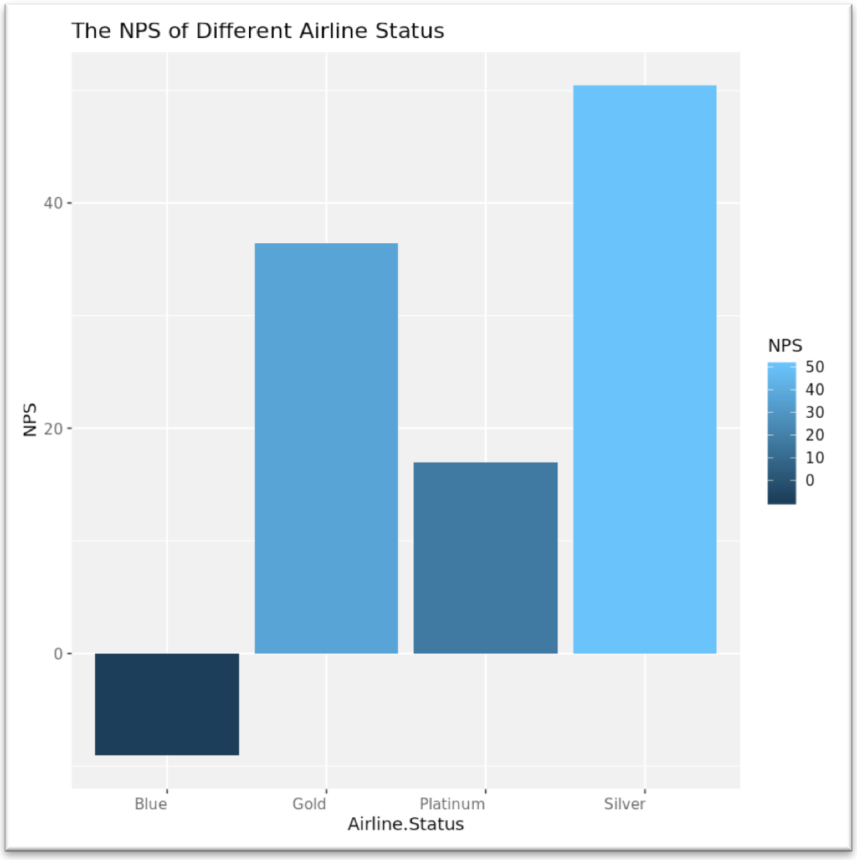
```
gender_compare <- df %>% group_by(Gender)%>%
  summarise(n=n(),promoter_count=length(type[type == "promoter"]),
            detractor_count=length(type[type == "detractor"]),
            NPS = (promoter_count/n - detractor_count/n)*100)

gender_plot <- gender_compare %>% ggplot(aes(x = Gender, y = NPS, fill = NPS)) +
  theme(axis.text.x = element_text(angle = 0, hjust = 1)) +
  ggtitle("The NPS of Male and Female")+
  geom_col()

class_compare <- df %>% group_by(Class)%>%
```

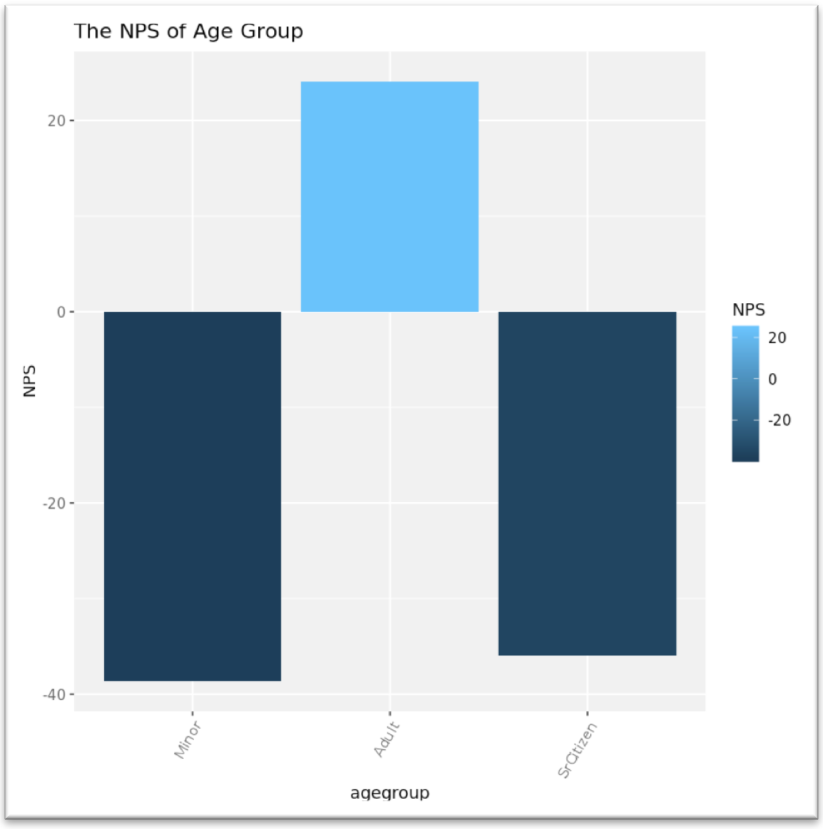
```
  summarise(n=n(),promoter_count=length(type[type == "promoter"]),
            detractor_count=length(type[type == "detractor"]),
            NPS = (promoter_count/n - detractor_count/n)*100)

class_plot <- class_compare %>% ggplot(aes(x = Class, y = NPS, fill = NPS)) +
  theme(axis.text.x = element_text(angle = 0, hjust = 1)) +
  ggtitle("The NPS of Different Class")+
  geom_col()

price_compare <- df %>% group_by(Price.Sensitivity)%>%
  summarise(n=n(),promoter_count=length(type[type == "promoter"]),
            detractor_count=length(type[type == "detractor"]),
            NPS = (promoter_count/n - detractor_count/n)*100)

price  compare %>% ggplot(aes(x = Price.Sensitivity, y = NPS, fill = NPS)) +
  theme(axis.text.x = element_text(angle = 0, hjust = 1)) +
  ggtitle("The NPS of Different Price.Sensitivity")+
  geom_col()

status_compare <- df %>% group_by(Airline.Status)%>%
  summarise(n=n(),promoter_count=length(type[type == "promoter"]),
            detractor_count=length(type[type == "detractor"]),
            NPS = (promoter_count/n - detractor_count/n)*100)

status  compare %>% ggplot(aes(x = Airline.Status, y = NPS, fill = NPS)) +
  theme(axis.text.x = element_text(angle = 0, hjust = 1)) +
  ggtitle("The NPS of Different Airline Status")+
  geom_col()

age_compare <- df %>% group_by(agegroup)%>%
  summarise(n=n(),promoter_count=length(type[type == "promoter"]),
            detractor_count=length(type[type == "detractor"]),
            NPS = (promoter_count/n - detractor_count/n)*100)

age_compare %>% ggplot(aes(x = agegroup , y = NPS,fill = NPS)) +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  ggtitle("The NPS of Age Group")+
  geom_col()

travel_compare <- df %>% group_by(Type.of.Travel)%>%
  summarise(n=n(),promoter_count=length(type[type == "promoter"]),
            detractor_count=length(type[type == "detractor"]),
            NPS = (promoter_count/n - detractor_count/n)*100)

travel_plot <- travel_compare %>% ggplot(aes(x = Type.of.Travel , y = NPS,fill = NPS))
+
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  ggtitle("The NPS of Type of Travel")+
  geom_col()
```

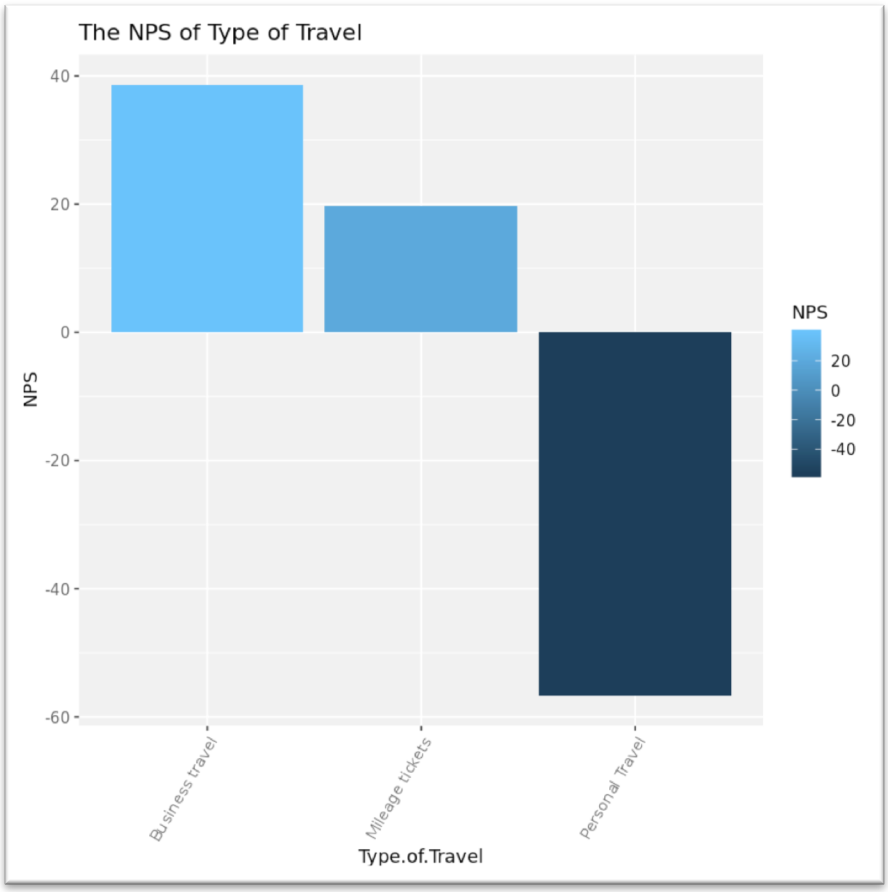

The NPS of Male and Female
15
10
5
0
NPS
Female
Male
Gender
NPS
15
10
5
0


The NPS of Different Class
20
15
10
5
0
NPS
Business
Eco
Eco Plus
Class
NPS
15
10
5


The NPS of Different Price.Sensitivity
10
0
-10
-20
-30
NPS
0
1
2
3
4
Price.Sensitivity
NPS
10
0
-10
-20
-30


The NPS of Different Airline Status
40
20
0
NPS
Blue
Gold
Platinum
Silver
Airline.Status
NPS
50
40
30
20
10
0


The NPS of Age Group
20
0
-20
-40
NPS
Minor
Adult
SrCitizen
agegroup
NPS
20
0
-20


The NPS of Type of Travel
40
20
0
-20
-40
-60
NPS
Business travel
Mileage tickets
Personal Travel
Type.of.Travel
NPS
20
0
-20
-40

***Insight**: According to the graphs, female travellers, Eco and Eco Plus travellers, blue airline status travellers, personal travellers, minor and senior citizens (age above 60 or below 18), and travellers have high sensitivity towards price are less likely to recommend the flights.*

5.2. TEXT MINING

In this section, we compare the key words in promoter and detractor travellers' comments on the Southeast airlines. The comments are stored in the 'freetext' column of the dataset.

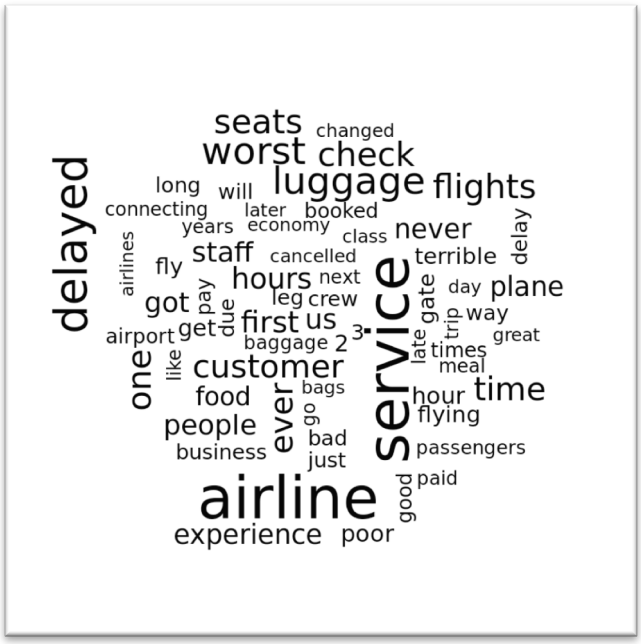
First, we need to create a stopword list to remove the most common words from the text. We use the TM package's default English stopword list and add 'southeast' and 'flight' to it.

```
# library relevant packages
library(tidyverse)
library(tm)
library(wordcloud)
library(tidytext)
# add a few words to the default stopword list
stop_words <- tibble(word = c(stopwords("english"),"flight","southeast"))
```

Then, we use the tidytext package's ''unnest_tokens' function to tokenize the 'freeText' into words, remove stopwords, create a word frequency table, and generate two separate word clouds for promoter words and detractor words:

```
df_pro <- df %>%
  filter(type == "promoter") %>%
  unnest_tokens(word, freeText) %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE)

wordcloud(df_pro$word, df_pro$n, min.freq = 10, ordered.colors= TRUE)

df_det <- df %>%
  filter(type == "detractor") %>%
  unnest_tokens(word, freeText) %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE)
```

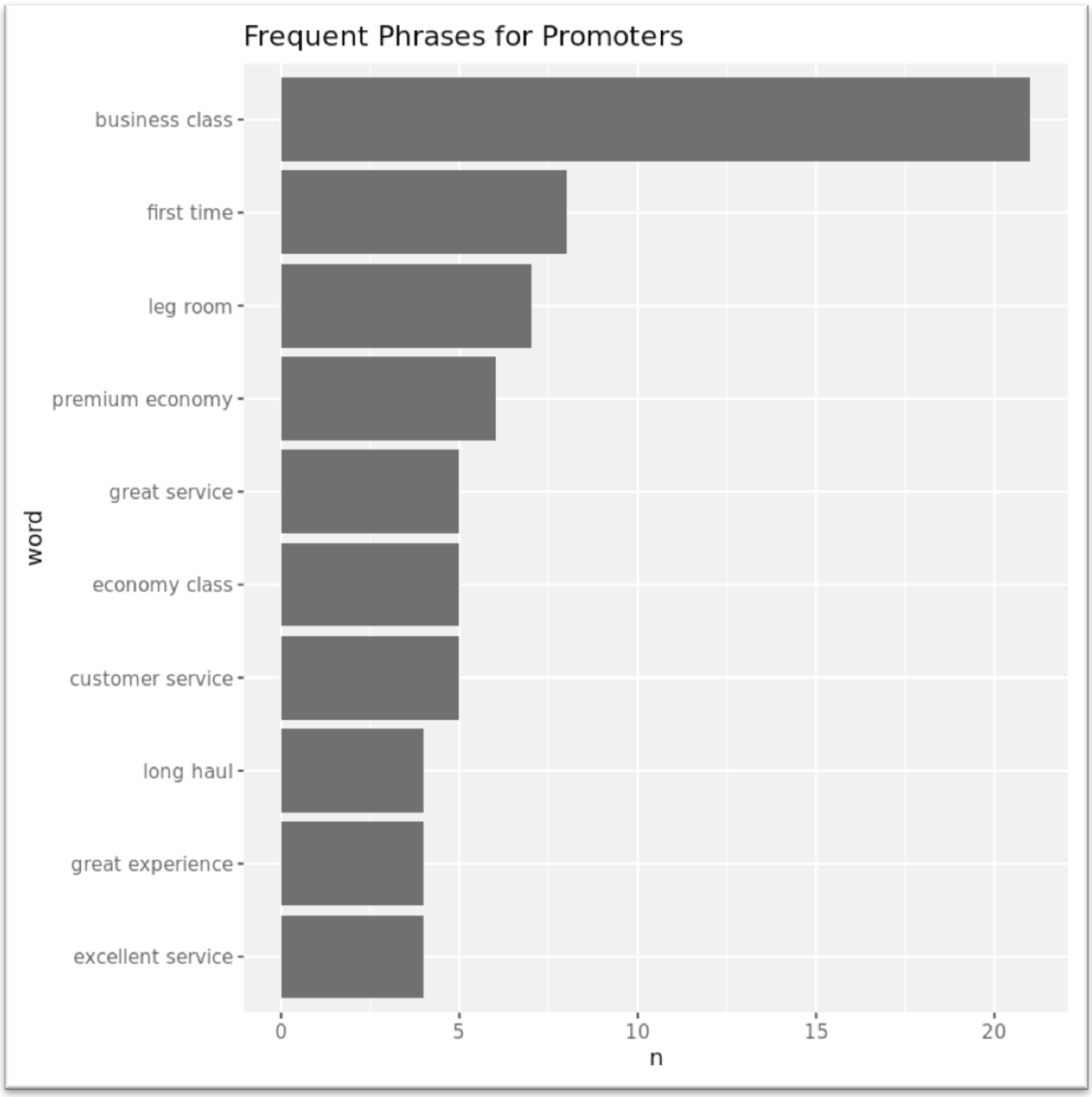
wordcloud(df_det$word, df_det$n, min.freq = 10, ordered.colors= TRUE)

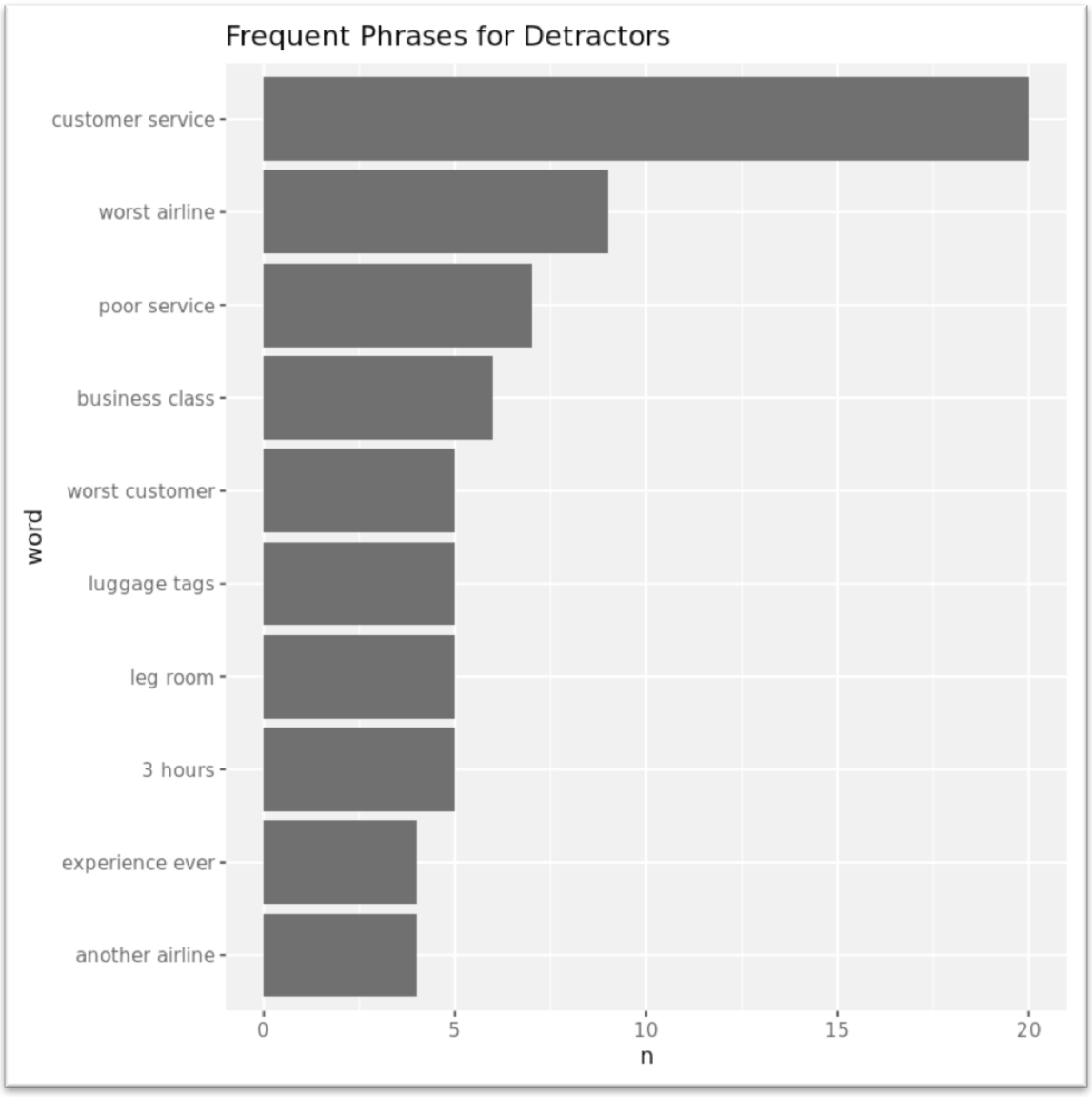


Insight*: As two graphs suggest, detractor customers tend to complain about the delay, service, and seats, while promoter customers praise about food and service. Although the word clouds provide some information of how travellers think of the airlines , it is still difficult to understand what words like 'class', 'customer', and 'luggage' are referring to.*

Therefore, we tokenize the text into bigrams (two adjacent words) to explore the key phrases in the customer comments. To remove the stopwords from the bigrams, we first separate the bigrams into words in two columns, filter the words in stopword list separately, and combine two words back together. Then we draw two bar charts for the most frequent two-word phrases in detractors and promoters.

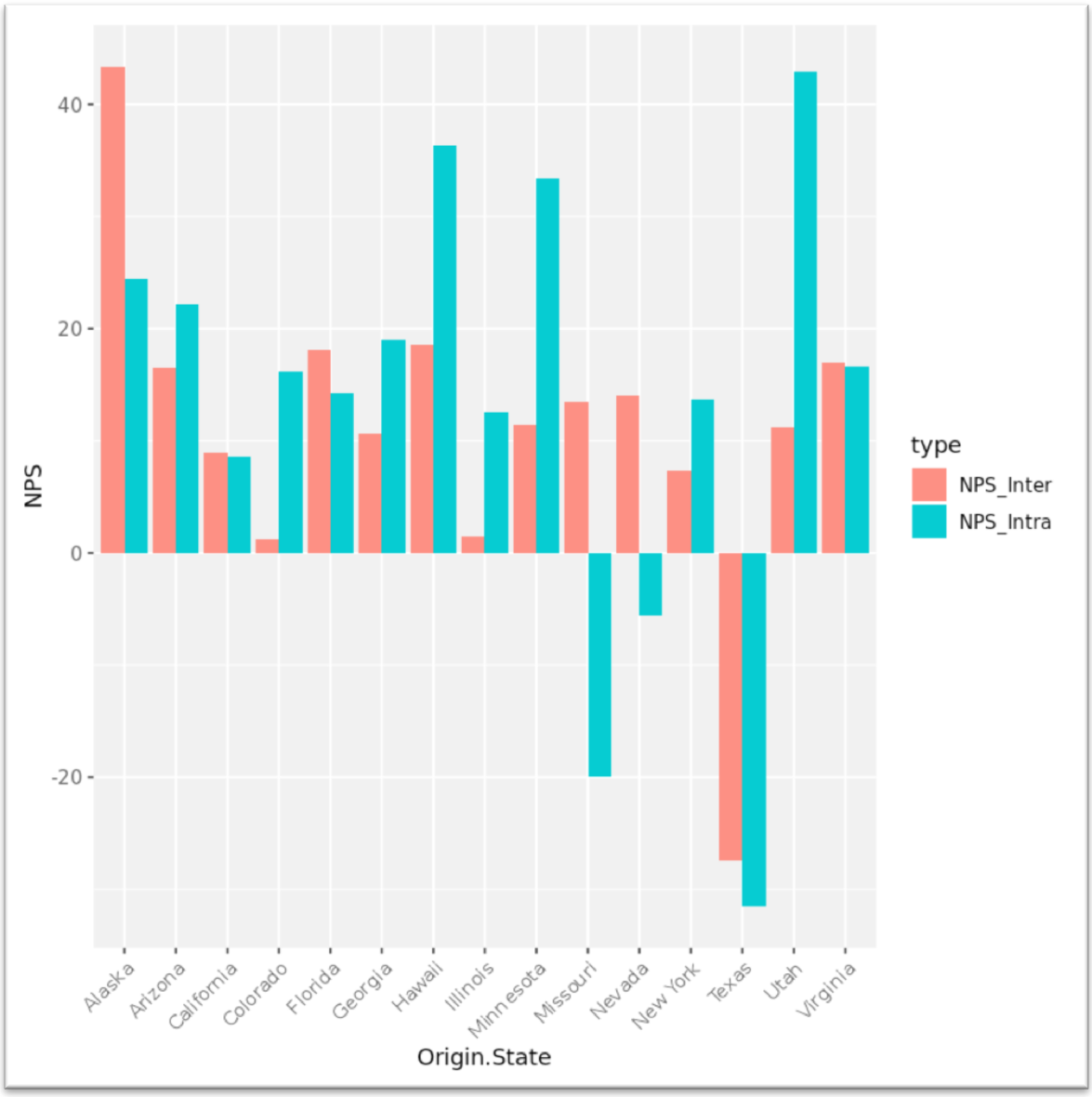
```
df %>%
 filter(type == "promoter") %>%
 unnest_tokens(word, freeText, token = "ngrams", n = 2) %>%
 separate(word, c("word1", "word2"), sep = " ") %>%
 filter(!word1 %in% stop_words$word) %>%
 filter(!word2 %in% stop_words$word) %>%
 unite(word,word1, word2, sep = " ")%>%
 count(word, sort =TRUE) %>%
 filter(n > 3) %>%
 mutate(word = reorder(word, n)) %>%
 ggplot(aes(x = word, y=n))+
 geom_col()+
 coord_flip()+
 ggtitle("Frequent Phrases for Promoters")

df %>%
 filter(type == "detractor") %>%
 unnest_tokens(word, freeText, token = "ngrams", n = 2) %>%
 separate(word, c("word1", "word2"), sep = " ") %>%
 filter(!word1 %in% stop_words$word) %>%
 filter(!word2 %in% stop_words$word) %>%
 unite(word,word1, word2, sep = " ")%>%
 count(word, sort =TRUE) %>%
 filter(n > 3) %>%
 mutate(word = reorder(word, n)) %>%
 ggplot(aes(x = word, y=n))+
 geom_col()+
 coord_flip()+
 ggtitle("Frequent Phrases for Detractors")
```


Frequent Phrases for Promoters
business class
first time
leg room
premium economy
great service
economy class
customer service
long haul
great experience
excellent service
word
0
5
10
15
20
n



***Insight**: As we can see from the graphs, unsatisfied customers are complaining about issues like 'leg room', 'luggage tags', and poor 'customer service', describing the airline as the 'worst airline', and planning to take 'another airline' in the future.*

5.3. INTER & INTRA STATE



***Insight**: As we can see from the graph, only Texas has negative NPS for both the interstate and intrastate routes. Missouri and Nevada have positive NPS for interstate routes but have negative NPS for intrastate routes.*

5.4. MAPS

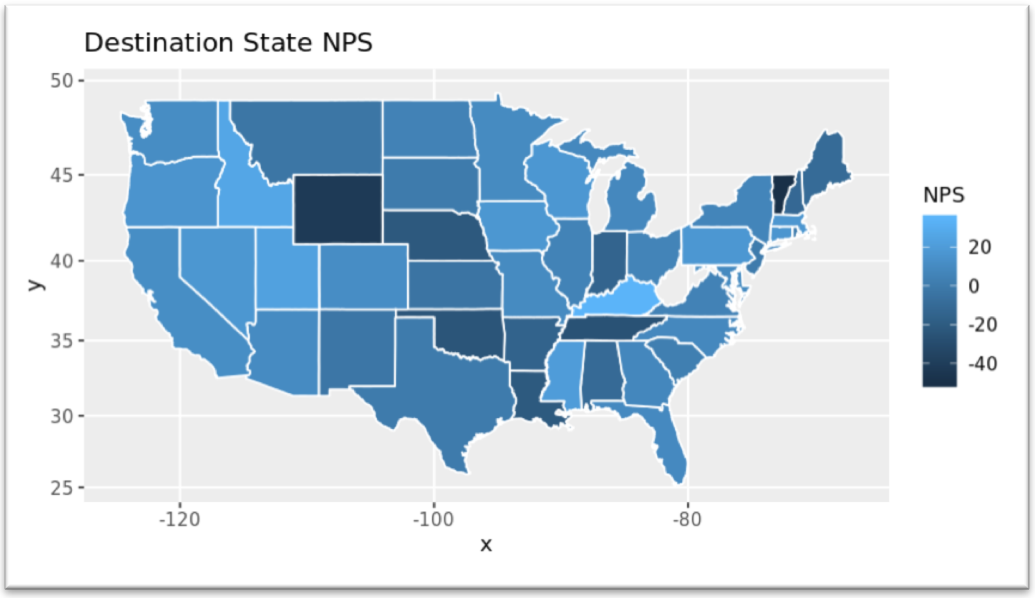
We were quite intrigued of how the customer NPS distribution varied with respect to geography. Therefore, we tried to work with the data at hand. We were forturnate that we had longitude and latitude coordinates of destination and origin cities and therefore didn't require any other package.

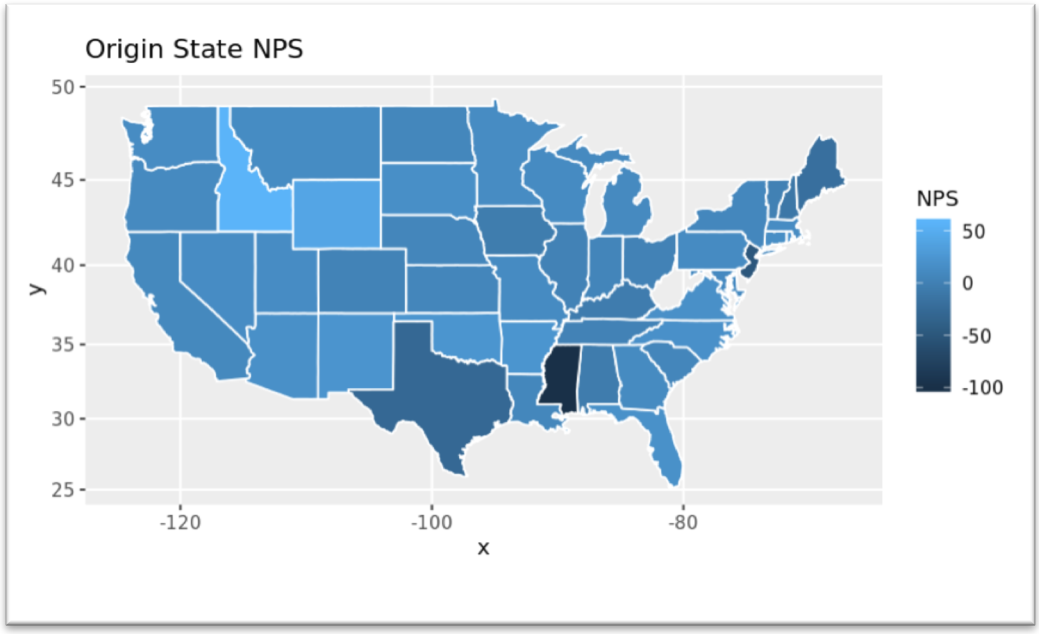
We started with finding some connection between NPS and destination/origin state.

Code:

```
states <- map_data("state")
df_states_dest <- df %>% group_by(Destination.State)%>%
  summarise(n=n(),promoter_count=length(type[type == "promoter"]),
            detractor_count=length(type[type == "detractor"]),
            NPS = (promoter_count/n - detractor_count/n)*100
  )%>%
  mutate(region=tolower(Destination.State))%>%
  inner_join(states, by = "region")
df_states_dest_plot <- df_states_dest %>%
  mutate(Destination.State=tolower(Destination.State))%>%
  mutate(NPS_type = cut(NPS,
                        breaks = c(-Inf,0,Inf),
                        labels = c("Detractor","Promoter"))) %>%
  ggplot(aes(map_id = Destination.State, label = Destination.State)) + geom_map(map =
states, aes(fill = NPS),color = "white")+ expand_limits(x = states$long, y = states$lat)+
coord_map() + ggtitle('Destination State NPS')
df_states_ori <- df %>% group_by(Origin.State)%>%
  summarise(n=n(),promoter_count=length(type[type == "promoter"]),
            detractor_count=length(type[type == "detractor"]),
            NPS = (promoter_count/n - detractor_count/n)*100)
df_states_ori_plot <- df_states_ori %>%
  mutate(Origin.State=tolower(Origin.State))%>%
  mutate(NPS_type = cut(NPS,
                        breaks = c(-Inf,0,Inf),
                        labels = c("Detractor","Promoter"))) %>%
  ggplot(aes(map_id = Origin.State, label = Origin.State)) + geom_map(map = states,
aes(fill = NPS),color = "white")+ expand_limits(x = states$long, y = states$lat)+
coord_map() + ggtitle('Origin State NPS')
```

Output:





We had some issues with this concept of working with states in general. It does not let us know what exactly goes on inside that state. Is there any problem with one particular airport or is there any intra state issues. There is an ambiguity with what we propose as well. Also we wanted to perform some quality checks for the output when we came across the following:

Destination.State	n	promoter_count	detractor_count	NPS
<chr>	<int>	<int>	<int>	<dbl>
Vermont	4	0	2	-50.000000
Mississippi	5	2	1	20.000000
New Jersey	6	2	2	0.000000
New Hampshire	17	5	7	-11.764706
Maine	20	5	7	-10.000000
Wyoming	21	4	13	-42.857143
North Dakota	22	9	8	4.545455
Idaho	23	10	4	26.086957
Kansas	23	8	9	-4.347826

Origin.State	n	promoter_count	detractor_count	NPS
<chr>	<int>	<int>	<int>	<dbl>
Mississippi	4	0	4	-100.000000
New Jersey	4	1	3	-50.000000
Maine	5	0	1	-20.000000
Vermont	6	2	2	0.000000
Idaho	7	5	1	57.142857
New Hampshire	9	3	4	-11.111111
Rhode Island	14	6	3	21.428571
North Dakota	15	4	3	6.666667
Kansas	17	5	4	5.882353
South Dakota	19	8	5	15.789474
Iowa	21	5	6	-4.761905

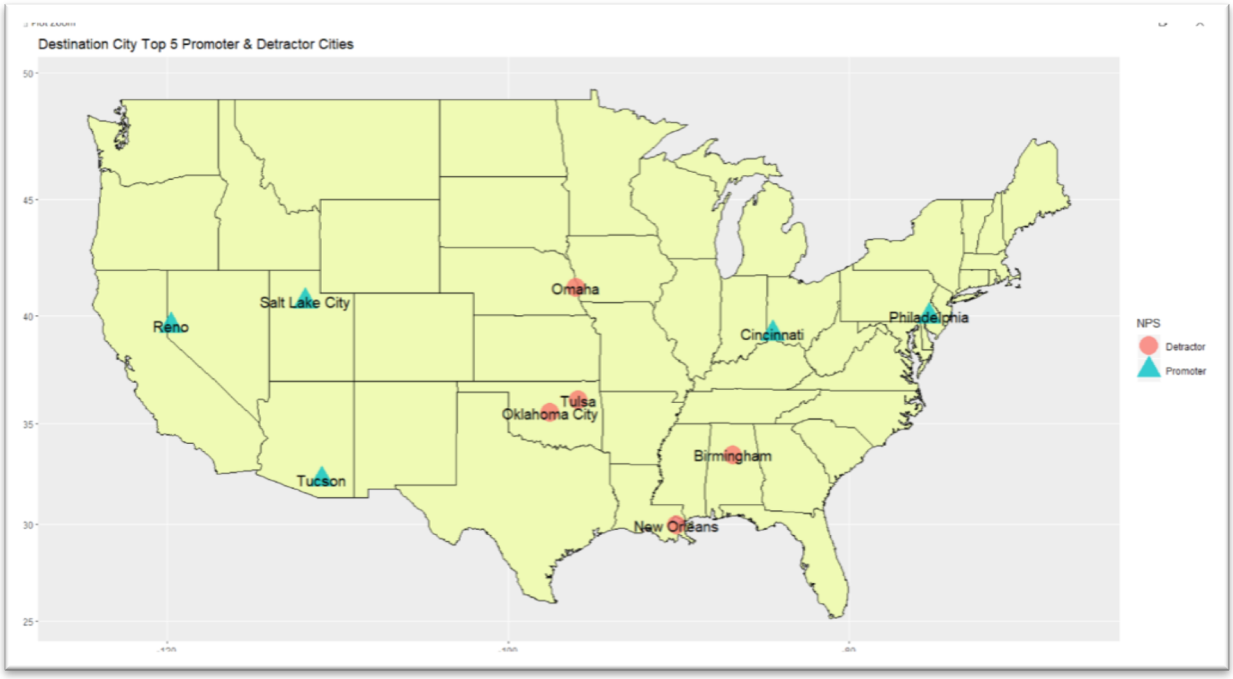
In terms of destination state, if we look at Vermont or Mississippi and the NPS score then we might feel that Vermont is bad and Mississippi is very good. But, if we look at the samples pertaining to the state we have is like 0.001% of the overall dataset and therefore we can't generalize the recommendation for all the states.

Similarly, In terms of origin state, Mississippi and Idaho have fewer flights flying from that state. Therefore, we decided not to consider this visualization. Another solution would've been removing the states which had fewer flights but then in terms of map

visualization it was looking bad and therefore, we decided not to do that and focus on City Visualizations and not the state.

5.4.1. Destination City NPS:

To tackle the problem which we faced during grouping states, we applied a filter for the number of flights taking off or arriving at an airport. We used trial and error filters and settled for n > 25 which means we are considering only those airport cities which have more than 25 flights leaving or arriving in that city.



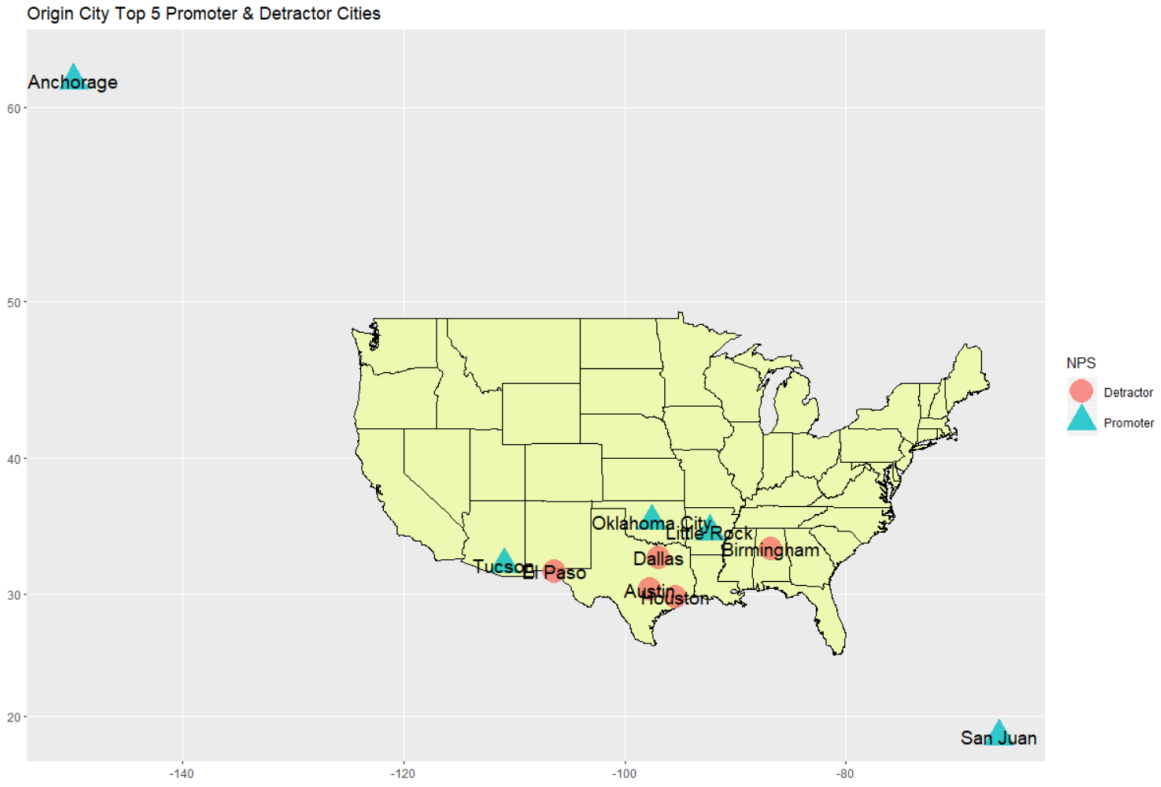
***Insight**: From the states, we focused our visualizations on Cities. We wanted to find out which airports are performing better or worse in terms of NPS. The red circles indicate the top 5 worst performing airport cities and green triangles represent the best 5 performing airport cities.*

Code:

```
# Grouping by destination city and then calculating NPS value for each city.
# Only considering those cities which had more than 25 flights flying in the dataset.
# DESTINATION CITY PLOT

df_city_des <- df %>% group_by(Destination.City, dlong, dlat)%>%
  summarise(n=n(),promoter_count=length(type[type == "promoter"]),
            detractor_count=length(type[type == "detractor"]),
```

```
        NPS = (promoter_count/n - detractor_count/n)*100
 ) %>%
 mutate(NPS_type = cut(NPS,
                       breaks = c(-Inf,0,Inf),
                       labels = c("Detractor","Promoter"))) %>%
 filter(n > 25) %>%
 arrange(NPS)
# Selecting only best 5 and worst five cities
df_city_des <- df_city_des[-6:-51,]
# Plot
dest_city_plot <- ggplot() +
 geom_polygon(data = states, color="black", fill= "#edf8b1",
              aes(x=long,y=lat, group=group)) +
 geom_point(data = df_city_des, aes(x = dlong, y = dlat, colour =
factor(NPS_type),shape = factor(NPS_type)),size =8,alpha = 0.8) + ggtitle("Destination
City Top 5 Promoter & Detractor Cities") + theme(axis.title.x = element_blank(),
axis.title.y = element_blank())+ labs(colour = "NPS", shape = "NPS") + coord_map() +
geom_text(data = df_city_des,aes(x = dlong, y = dlat, label = Destination.City, group =
NULL), size = 5)
```

5.4.2. Origin City NPS:



***Insight**: We observed a few things from these visualizations. Tucson is consistently a top performer in terms of NPS for the destination as well as arriving cities. While Oklahoma City is one of the top promoters for Origin City flights, it is one of the worst detractors in*

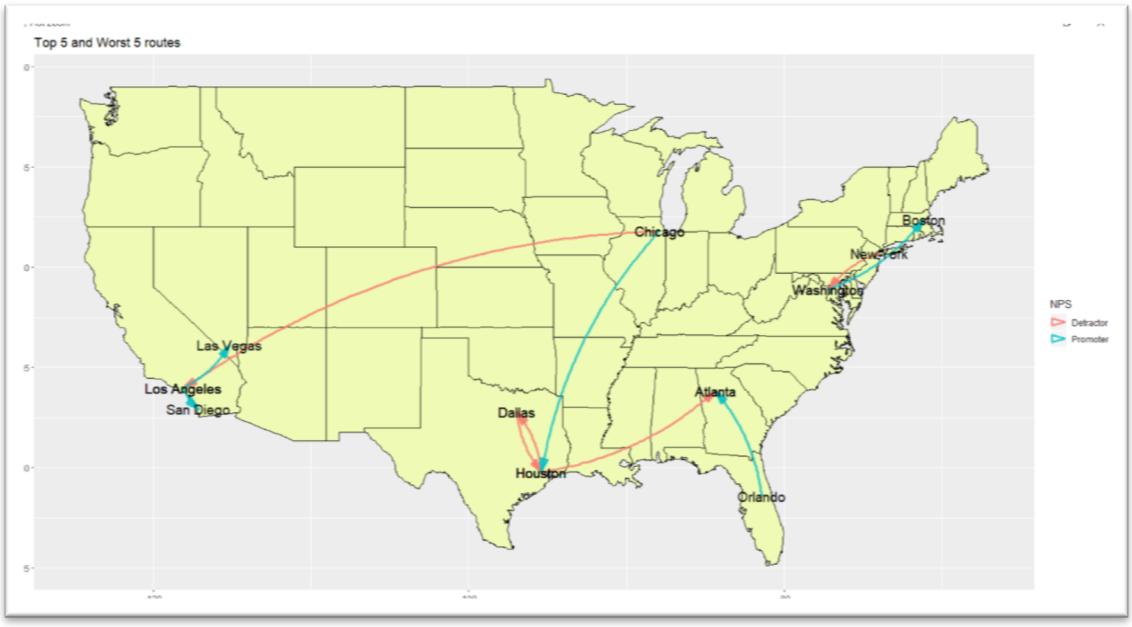
terms of Destination city flights. And Birmingham is a poor performer for both Destination City as well as Arriving City flights.

Code:

```
# Grouping by origin city and then calculating NPS value for each city.
# Only considering those cities which had more than 25 flights flying in the dataset.
# ORIGIN CITY PLOT

df_city_arr <- df %>% group_by(Origin.City, olong, olat)%>%
  summarise(n=n(),promoter_count=length(type[type == "promoter"]),
            detractor_count=length(type[type == "detractor"]),
            NPS = (promoter_count/n - detractor_count/n)*100
  ) %>%
  mutate(NPS_type = cut(NPS,
                        breaks = c(-Inf,0,Inf),
                        labels = c("Detractor","Promoter"))) %>%
  filter(n > 25) %>%
  arrange(NPS)
# Selecting only best 5 and worst five cities
df_city_arr <- df_city_arr[-6:-51,]
# PLOT
ori_city_plot <- ggplot() +
  geom_polygon(data = states, color="black", fill= "#edf8b1",
               aes(x=long,y=lat, group=group)) +
  geom_point(data = df_city_arr, aes(x = olong, y = olat, colour = factor(NPS_type),shape = factor(NPS_type)),size =8,alpha = 0.8) + ggtitle("Origin City Top 5 Promoter & Detractor Cities") + theme(axis.title.x = element_blank(), axis.title.y = element_blank())+ labs(colour = "NPS", shape = "NPS") + coord_map() + geom_text(data = df_city_arr,aes(x = olong, y = olat, label = Origin.City, group = NULL), size = 5)
```

5.4.3. Route:

We wanted to go one step further by not restricting ourselves just on identifying which cities are performing good or bad. We wanted to look at routes as well. Therefore, we made a group of Origin City and Destination City and started looking at it from NPS point of view. Even here we used filter to not consider data which was sparsely distributed. Green coloured routes are best performing routes and red coloured routes indicate worst performing routes.



***Insight**: Here, we observed that Dallas – Houston and Houston – Dallas are poor performing routes in terms of NPS. And there has to be something inside the airport or flights getting cancelled because of which people are giving very low NPS scores.*

Code:

```
  df_route <- df %>% group_by(Origin.City, Destination.City, olong, olat, dlong, dlat)%>%
 summarise(n=n(),promoter_count=length(type[type == "promoter"]),
           detractor_count=length(type[type == "detractor"]),
           NPS = (promoter_count/n - detractor_count/n)*100) %>%
 mutate(NPS_type = cut(NPS,
                       breaks = c(-Inf,0,Inf),
                       labels = c("Detractor","Promoter"))) %>%
 filter(n>25)%>%
 arrange(NPS)
# Selecting only best 5 and worst five cities
df_route <- df_route[-6:-25,]
# Storing as dataframe
df_route <- as.data.frame(df_route)
# Route plot
route_plot<- ggplot() + geom_polygon(data = states, color="black", fill= "#edf8b1",aes(x=long,y=lat, group=group)) + geom_curve(data = df_route ,aes(x = olong, y = olat, xend = dlong, yend = dlat, colour = factor(NPS_type)), arrow = arrow(angle = 15, length = unit(0.5, "cm"), type = "closed"), size = 1.1, alpha = 0.8, curvature = 0.15, inherit.aes = TRUE) +  coord_map() + coord_cartesian() + ggtitle("Top 5 and Worst 5 routes") + theme(axis.title.x = element_blank(), axis.title.y = element_blank()) + geom_text(data = df_route,aes(x = olong, y = olat, label = Origin.City, group = NULL, check_overlap = TRUE), size = 5) + geom_text(data = df_route,aes(x = dlong, y = dlat, label = Destination.City, group = NULL, check_overlap = TRUE), size = 5) + labs(colour = "NPS", shape = "NPS")
```

6. MODELS

6.1. LINEAR MODELS

We used multivariate models and bivariate plots using all the significant continuous variables to understand and predict patterns for different combinations. The plots especially helped us define the relationship of different variables with likelihood to recommend. For the models we first combined all the continuous variables which resulted into R-squared value 0.09.

CODE :

```
model7<- lm(formula = Likelihood.to.recommend ~ Loyalty + Total.Freq.Flyer.Accts + Age + Price.Sensitivity + Flights.Per.Year + ArrivalDelayRatio+ DepartureDelayRatio , data= df)
summary(model7)
```

OUTPUT :

```
Residuals:
    Min      1Q  Median      3Q     Max
-7.3406 -1.3484  0.4689  1.6412  4.7884

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           9.692998   0.098581  98.326  < 2e-16 ***
Loyalty              -0.225825   0.060113  -3.757 0.000173 ***
Total.Freq.Flyer.Accts -0.059671   0.021806  -2.736 0.006222 **
Age                  -0.025544   0.001403 -18.205  < 2e-16 ***
Price.Sensitivity    -0.412969   0.039418 -10.477  < 2e-16 ***
Flights.Per.Year     -0.035861   0.002151 -16.675  < 2e-16 ***
ArrivalDelayRatio    -0.860010   0.166188  -5.175 2.32e-07 ***
DepartureDelayRatio   0.815821   0.161795   5.042 4.68e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.167 on 10253 degrees of freedom
Multiple R-squared:  0.09348,   Adjusted R-squared:  0.09286
F-statistic:   151 on 7 and 10253 DF,  p-value: < 2.2e-16
```

Further, we combined various variables, generated models and observed the significance of each variable with respect to the variables it is compared with and also the R-square value.

CODE :

model1<- lm(formula = Likelihood.to.recommend ~ Age + Price.Sensitivity + Flights.Per.Year, data= df)
summary(model1)

OUTPUT :

```
Residuals:
   Min     1Q Median     3Q    Max
-7.314 -1.360  0.490  1.655  4.725

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        9.424839   0.083288  113.16   <2e-16 ***
Age               -0.023044   0.001288  -17.89   <2e-16 ***
Price.Sensitivity -0.396626   0.039330  -10.09   <2e-16 ***
Flights.Per.Year  -0.029664   0.001550  -19.14   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.173 on 10257 degrees of freedom
Multiple R-squared:  0.08794,   Adjusted R-squared:  0.08767
F-statistic: 329.6 on 3 and 10257 DF,  p-value: < 2.2e-16
```

CODE :

model2<- lm(formula = Likelihood.to.recommend ~ Loyalty + Price.Sensitivity + Flights.Per.Year, data= df)
summary(model2)

OUTPUT :

```
Residuals:
    Min      1Q  Median      3Q     Max
-7.2042 -1.3782  0.4755  1.6594  4.5424

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         8.46668    0.06563 129.005   <2e-16 ***
Loyalty            -0.12098    0.05802  -2.085   0.0371 *
Price.Sensitivity -0.34720    0.03993  -8.696   <2e-16 ***
Flights.Per.Year  -0.03971    0.00217 -18.301   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.206 on 10257 degrees of freedom
Multiple R-squared:  0.05987,   Adjusted R-squared:  0.05959
F-statistic: 217.7 on 3 and 10257 DF,  p-value: < 2.2e-16
```

CODE *:*

```
model3<- lm(formula = Likelihood.to.recommend ~ Loyalty + Total.Freq.Flyer.Accts + Flights.Per.Year, data= df)
summary(model3)
```

OUTPUT :

```
Residuals:
    Min      1Q  Median      3Q     Max
-6.9515 -1.4325  0.4806  1.6654  4.5809

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           7.927349   0.045475 174.323  < 2e-16 ***
Loyalty              -0.167890   0.061186  -2.744  0.00608 **
Total.Freq.Flyer.Accts  0.091272   0.020592   4.432 9.42e-06 ***
Flights.Per.Year     -0.039467   0.002179 -18.116  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.212 on 10257 degrees of freedom
Multiple R-squared:  0.05475,   Adjusted R-squared:  0.05447
F-statistic:   198 on 3 and 10257 DF,  p-value: < 2.2e-16
```

CODE :

```
model4<- lm(formula = Likelihood.to.recommend ~ Loyalty + Total.Freq.Flyer.Accts + Age, data= df)
summary(model4)
```

OUTPUT :

```
Residuals:
    Min      1Q  Median      3Q     Max
-7.3747 -1.4326  0.4831  1.7174  3.8968

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)               8.701677   0.078439 110.936  < 2e-16 ***
Loyalty                   0.496085   0.044327  11.192  < 2e-16 ***
Total.Freq.Flyer.Accts -0.089127   0.022080  -4.037 5.46e-05 ***
Age                       -0.026549   0.001414 -18.778  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.21 on 10257 degrees of freedom
Multiple R-squared:  0.05692,   Adjusted R-squared:  0.05665
F-statistic: 206.4 on 3 and 10257 DF,  p-value: < 2.2e-16

> |
```

CODE *:*

model5<- lm(formula = Likelihood.to.recommend ~ Age + Price.Sensitivity + Total.Freq.Flyer.Accts, data= df)

summary(model5)

OUTPUT :

```
Residuals:
    Min      1Q  Median      3Q     Max
-7.0057 -1.4055  0.4933  1.6857  4.6046

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)              9.20453    0.09724  94.661   <2e-16 ***
Age                     -0.02996    0.00141 -21.253   <2e-16 ***
Price.Sensitivity       -0.42082    0.04008 -10.498   <2e-16 ***
Total.Freq.Flyer.Accts -0.02947    0.02114  -1.394    0.163
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.211 on 10257 degrees of freedom
Multiple R-squared:  0.05556,   Adjusted R-squared:  0.05528
F-statistic: 201.1 on 3 and 10257 DF,  p-value: < 2.2e-16

> |
```

CODE *:*

model6<- lm(formula = Likelihood.to.recommend ~ ArrivalDelayRatio + DepartureDelayRatio, data= df)

summary(model6)

OUTPUT :

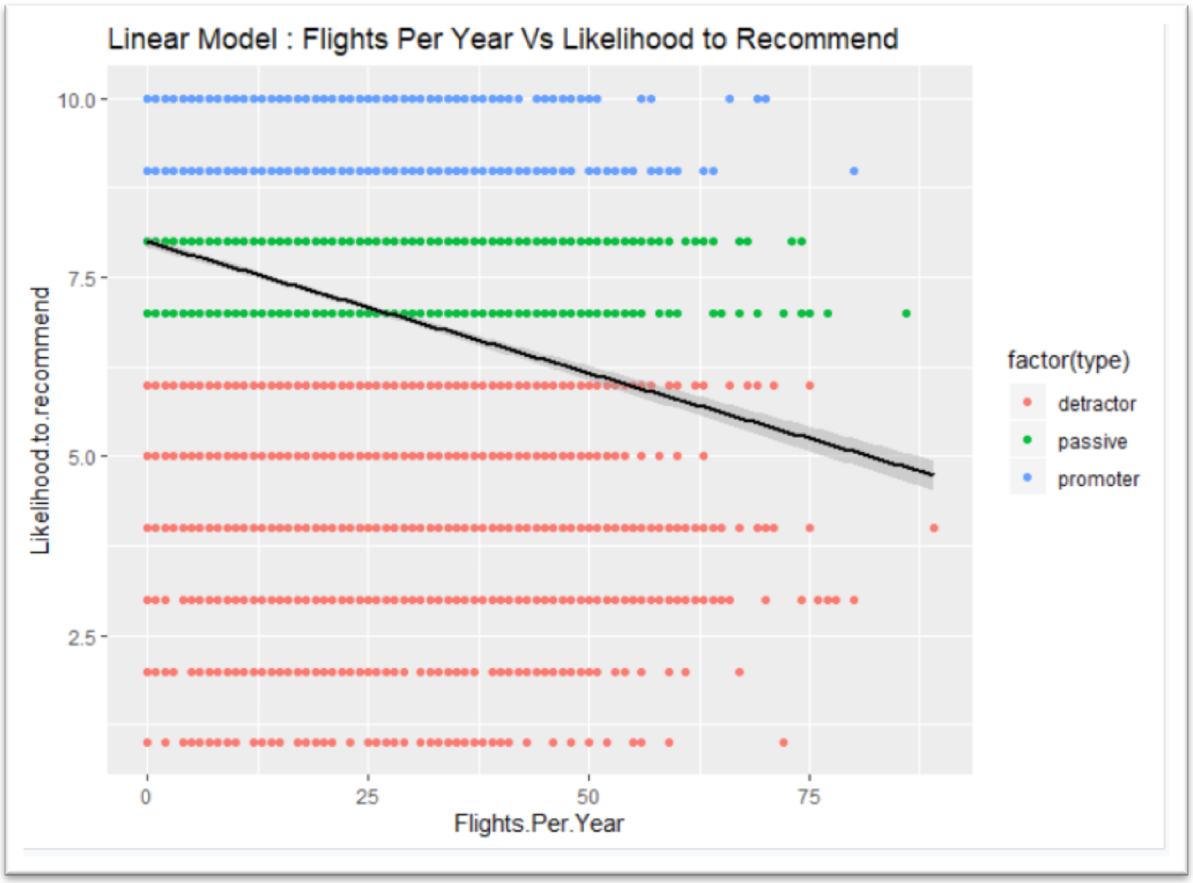
```
Residuals:
    Min      1Q  Median      3Q     Max
-6.4790 -1.2799  0.7201  1.7201  4.7866

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         7.27987    0.02283 318.926  < 2e-16 ***
ArrivalDelayRatio  -0.80536    0.17414  -4.625 3.80e-06 ***
DepartureDelayRatio 0.75575    0.16954   4.458 8.38e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.272 on 10258 degrees of freedom
Multiple R-squared:  0.003077,  Adjusted R-squared:  0.002883
F-statistic: 15.83 on 2 and 10258 DF,  p-value: 1.365e-07

> |
```

Majority of the models resulted in R-squared value to be 0.05, with highest 0.08 and lowest 0.002.

Then we plotted each variable with likelihood to recommend and generated some insights.

6.1.1. Flights per year vs Likelihood to recommend

CODE:
plot2<- ggplot(df, aes(y = Likelihood.to.recommend, x= Flights.Per.Year))
plot2<- plot2+ geom_point(aes(colour = factor(type))) + stat_smooth(method = "lm", color= "black")+ ggtitle("Linear Model : Flights Per Year Vs Likelihood to Recommend")
plot2



***Insight**: As customers take more number of flights the likelihood to recommend decreases.*

6.1.2. Loyalty vs Likelihood to recommend

***CODE**:*

```
plot3<- ggplot(df, aes(y = Likelihood.to.recommend, x= Loyalty))
plot3<- plot3+ geom_point(aes(colour = factor(type))) + stat_smooth(method = "lm",
color= "black")+ ggtitle("Linear Model : Loyalty Vs Likelihood to Recommend")
plot3
```

***Insight**: The more loyal a customers is the more chances that he or she is a promoter.*

6.1.3. Total frequent flyer accounts vs Likelihood to recommend

***CODE**:*

```
plot4<- ggplot(df, aes(y = Likelihood.to.recommend , x= Total.Freq.Flyer.Accts))
plot4<- plot4+ geom_point(aes(colour = factor(type))) +  geom_smooth(method = "lm",
color= "black")+ ggtitle("Linear Model : Total Frequent Flyer Accounts Vs Likelihood to
Recommend")
plot4
```

***Insight**: As customers make more number of flyer accounts they are more likely to be the promoters.*

6.1.4. Arrival delay ratio vs Likelihood to recommend

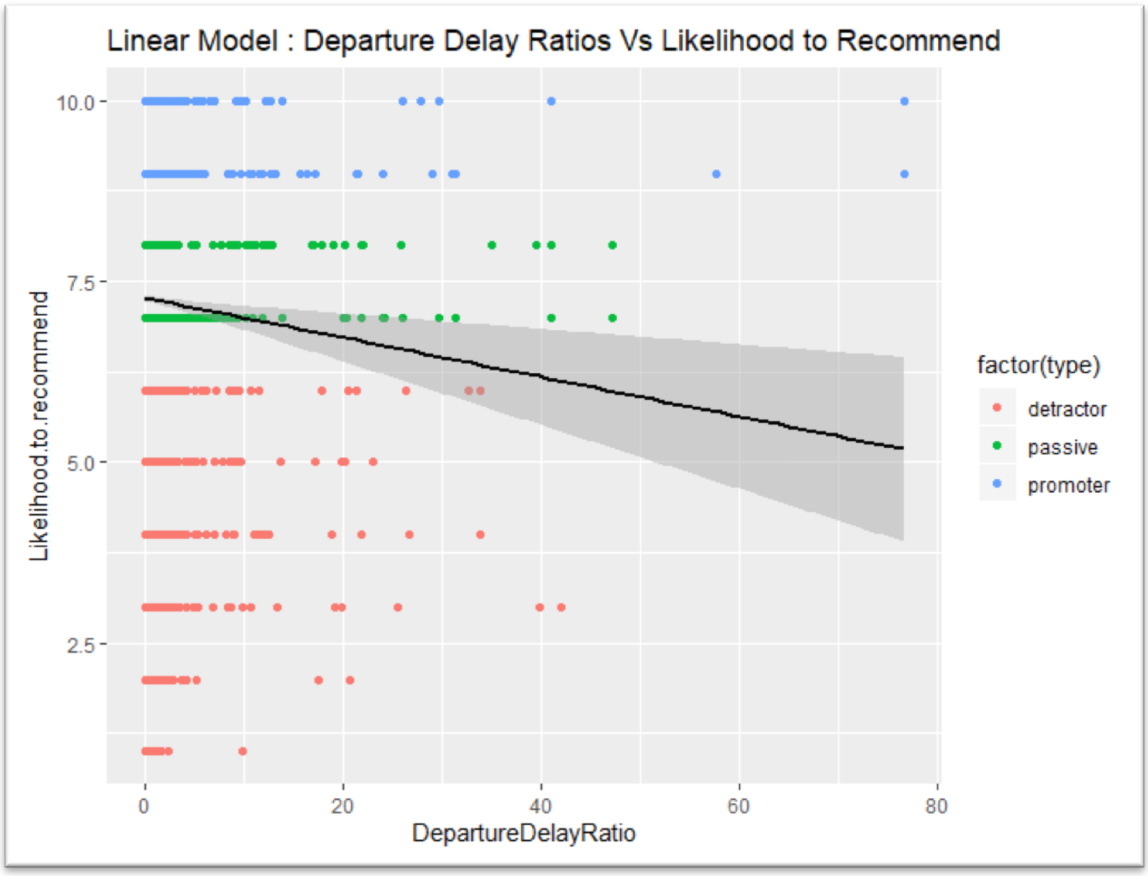
***CODE**:*
plot5<- ggplot(df, aes(y = Likelihood.to.recommend , x= ArrivalDelayRatio))
plot5<- plot5+ geom_point(aes(colour = factor(type))) + geom_smooth(method = "lm", color= "black")+ ggtitle("Linear Model : Arrival Delay Ratio Vs Likelihood to Recommend")
plot5

***Insight**: As the arrival delay ratio increases, likelihood to recommend goes low.*

6.1.5. Departure delay ratio vs Likelihood to recommend

***CODE**:*

```
plot6<- ggplot(df, aes(y = Likelihood.to.recommend , x= DepartureDelayRatio))
plot6<- plot6+ geom_point(aes(colour = factor(type))) + geom_smooth(method = "lm", color= "black")+ ggtitle("Linear Model : Departure Delay Ratios Vs Likelihood to Recommend")
plot6
```

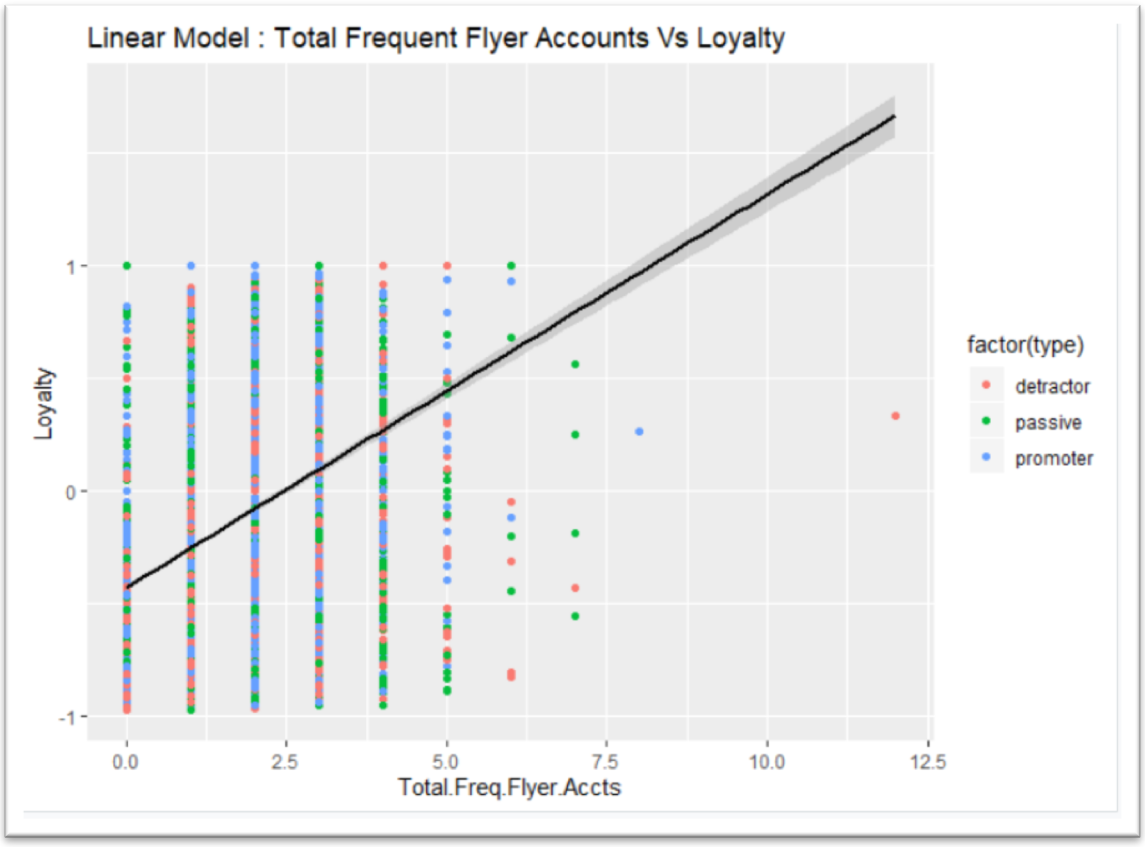


***Insight**: Similarly, more the departure delay, less is the likelihood to recommend score.*

We also defined a relationship between loyalty and total frequent flyer accounts and verified that as the total number of flyer accounts increases is the customer is more loyal.

***CODE** :*

```
plot1<- ggplot(df, aes(y = Loyalty, x= Total.Freq.Flyer.Accts ))
plot1<- plot1+ geom_point(aes(colour = factor(type))) + geom_smooth(method = "lm",
color= "black")+ ggtitle("Linear Model : Total Frequent Flyer Accounts Vs Loyalty")
plot1
```



In conclusion, we gathered many insights from the plots and models, but since the R-squared value was low, we could not base our analysis on just the linear models. Hence we continued to analyse the data with few more models.

6.2. LOGISTIC MODEL

The key concern of this project is to predict whether a passenger is a detractor or not, based on the data we have. As a classification model for categorical variables, logistic regression is suitable for this scenario. In this section, we built logistic regression model to evaluate how the variation of Class, Airline Status, and Type of Travel affect whether a passenger becomes a detractor or promoter.

First, we remove the 'passive' passengers from the data, since we are only interested in the classification of promoters and detractors. Meanwhile, to prepare the data for logistic regression, we create a new factor variable 'prob', which represents the probability of being a detractor. The 'prob' equals to 1 when the passenger is a detractor and it equals to 0 when he/she is a promoter.

```
df_classification <- df %>%
    # only include the promoter&detractor
 filter(type != 'passive')%>%
 mutate(prob = factor(ifelse(type == "detractor", 1, 0), levels = c(0,1))
```

Then, we divide the dataset into training data (80%) and test data (20%) using the caret package's createDataPartition function.

```
library(caret)

# create an index
set.seed(100) # set the seed for random number
index <- sample(1:6961, 6961) # generate a list of random number

# separate the data into train and test set
trainList <- createDataPartition(index,p=.8,list=FALSE)
train <- df_classification[trainList, ]
test <- df_classification[-trainList, ]
```

Next, we build a logistic regression model using the glm function and have a closer look at the model using the summary function.

```
model1 <- glm(prob ~ Airline.Status + Type.of.Travel + Class,
              family=binomial,
              data=train)
```

```
Call:
glm(formula = prob ~ Airline.Status + Type.of.Travel + Class,
    family = binomial, data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5022  -0.8630  -0.2305   0.3162   2.8191

Coefficients:
                             Estimate Std. Error z value Pr(>|z|)
(Intercept)                   -1.1355     0.1403  -8.093 5.84e-16 ***
Airline.StatusGold            -1.1397     0.1347  -8.458  < 2e-16 ***
Airline.StatusPlatinum        -0.3328     0.1678  -1.983   0.0474 *
Airline.StatusSilver          -2.8191     0.1407 -20.036  < 2e-16 ***
Type.of.TravelMileage tickets  0.5043     0.1256   4.014 5.98e-05 ***
Type.of.TravelPersonal Travel  3.7670     0.1069  35.244  < 2e-16 ***
ClassEco                       0.3396     0.1427   2.379   0.0174 *
ClassEco Plus                  0.4544     0.1797   2.529   0.0114 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7666.9  on 5568  degrees of freedom
Residual deviance: 4517.6  on 5561  degrees of freedom
AIC: 4533.6

Number of Fisher Scoring iterations: 5
```

As model summary suggests, the change of airline status to silver and gold are the most significant factors that make a passenger less likely to be a detractor, while choosing personal travel instead of Business travel is a significant factor in making a passenger detractor.

Now, we use the test data to evaluate the accuracy of the model. We label a passenger as a detractor when his/her probability of being a detractor is more than 0.5. As we can see from the result of the confusion matrix, the model performs well, reaching an accuracy of 0.817.

```
# predict
pred1 <- predict(model1, newdata = test, type = "response")

# test the accuracy
y_pred_num <- ifelse(pred1 > 0.5, 1, 0)
y_pred <- factor(y_pred_num, levels=c(0, 1))
y_act <- test$prob
```

```
# Confusion Matrix
table(y_pred, y_act)

# accuracy rate
mean(y_pred == y_act)
```

```
# Confusion Matrix
table(y_pred, y_act)
```

```
      y_act
y_pred   0   1
     0 744 201
     1  54 393
```

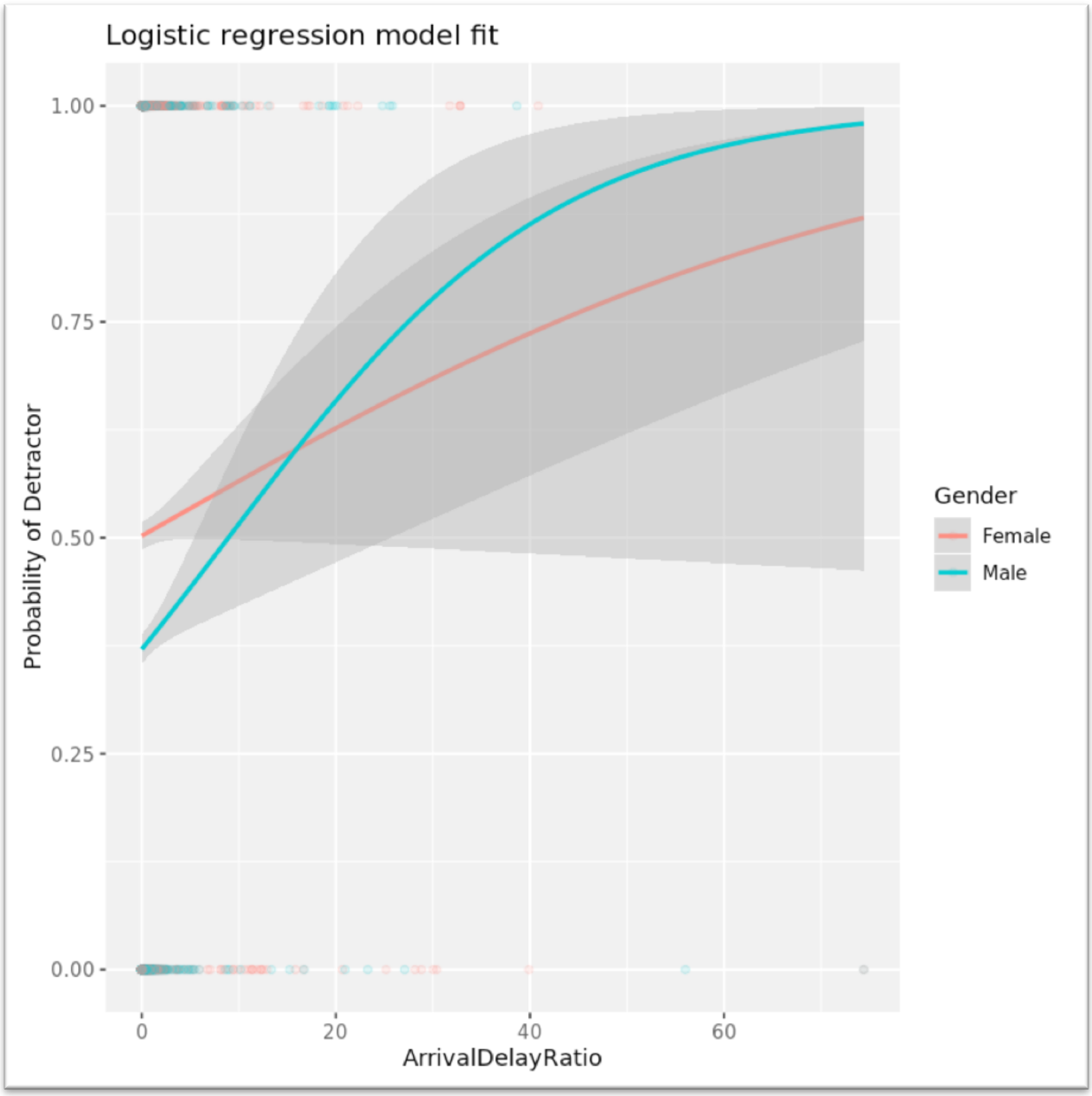
```
# accuracy rate
mean(y_pred == y_act)
```

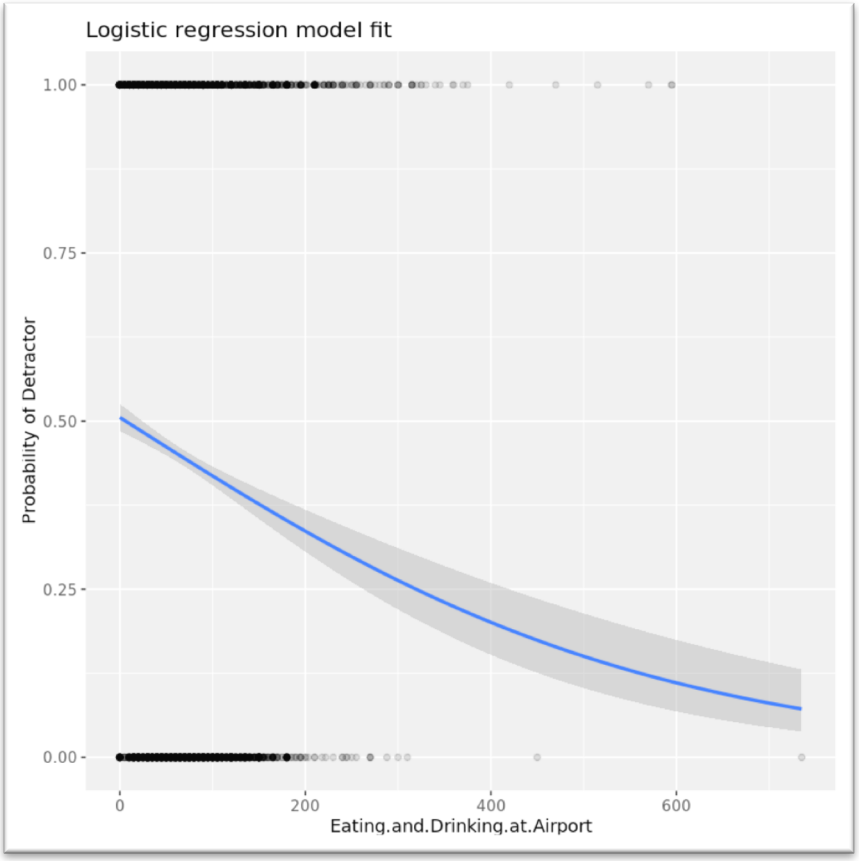
```
0.816810344827586
```

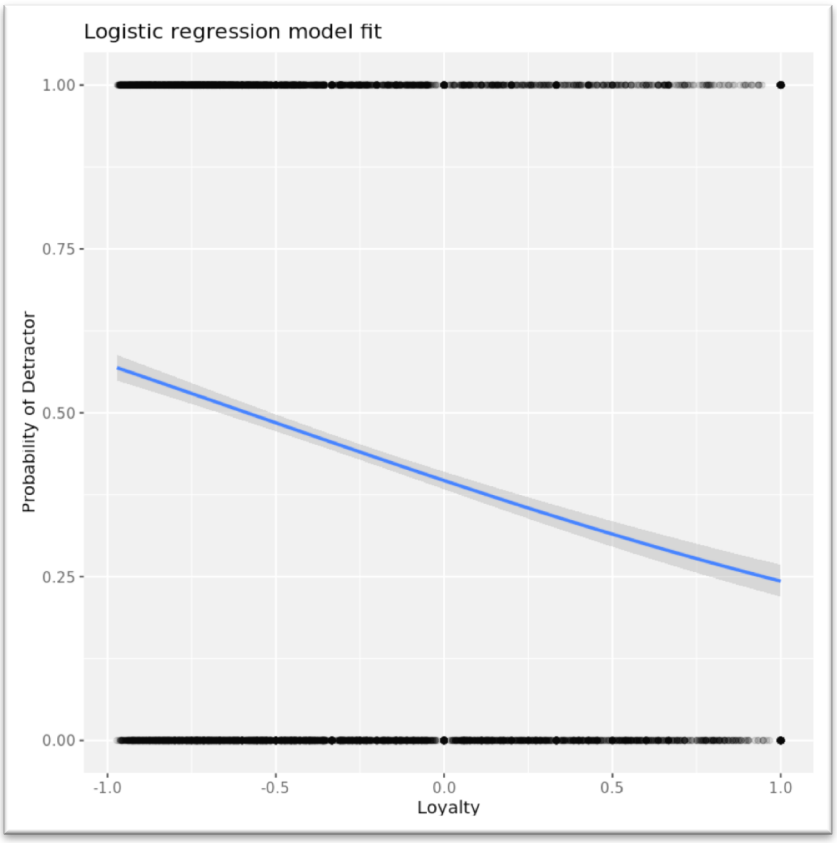
Meanwhile, we also plot the regression model for other variables, such as gender, departure delay, loyalty, and eating & drinking at the airport.

```
df_classification %>%
  ggplot(aes(x = DepartureDelayRatio, y= (as.numeric(prob)-1), color = Gender)) +
  geom_point(alpha = .15) +
  geom_smooth(method = "glm", method.args = list(family = "binomial")) +
  ggtitle("Logistic regression model fit") +
  xlab("DepartureDelayRatio") +
  ylab("Probability of Detractor")

df_classification %>%
  ggplot(aes(Eating.and.Drinking.at.Airport, (as.numeric(prob)-1))) +
  geom_point(alpha = .15) +
  geom_smooth(method = "glm", method.args = list(family = "binomial")) +
  ggtitle("Logistic regression model fit") +
  xlab("Eating.and.Drinking.at.Airport") +
  ylab("Probability of Detractor")

df_classification %>%
  ggplot(aes(Loyalty, (as.numeric(prob)-1))) +
  geom_point(alpha = .15) +
  geom_smooth(method = "glm", method.args = list(family = "binomial")) +
```

ggtitle("Logistic regression model fit") +
xlab("Loyalty") +
ylab("Probability of Detractor")




Logistic regression model fit
1.00
0.75
0.50
0.25
0.00
Probability of Detractor
0
200
400
600
Eating.and.Drinking.at.Airport


Logistic regression model fit
1.00
0.75
0.50
0.25
0.00
Probability of Detractor
-1.0
-0.5
0.0
0.5
1.0
Loyalty

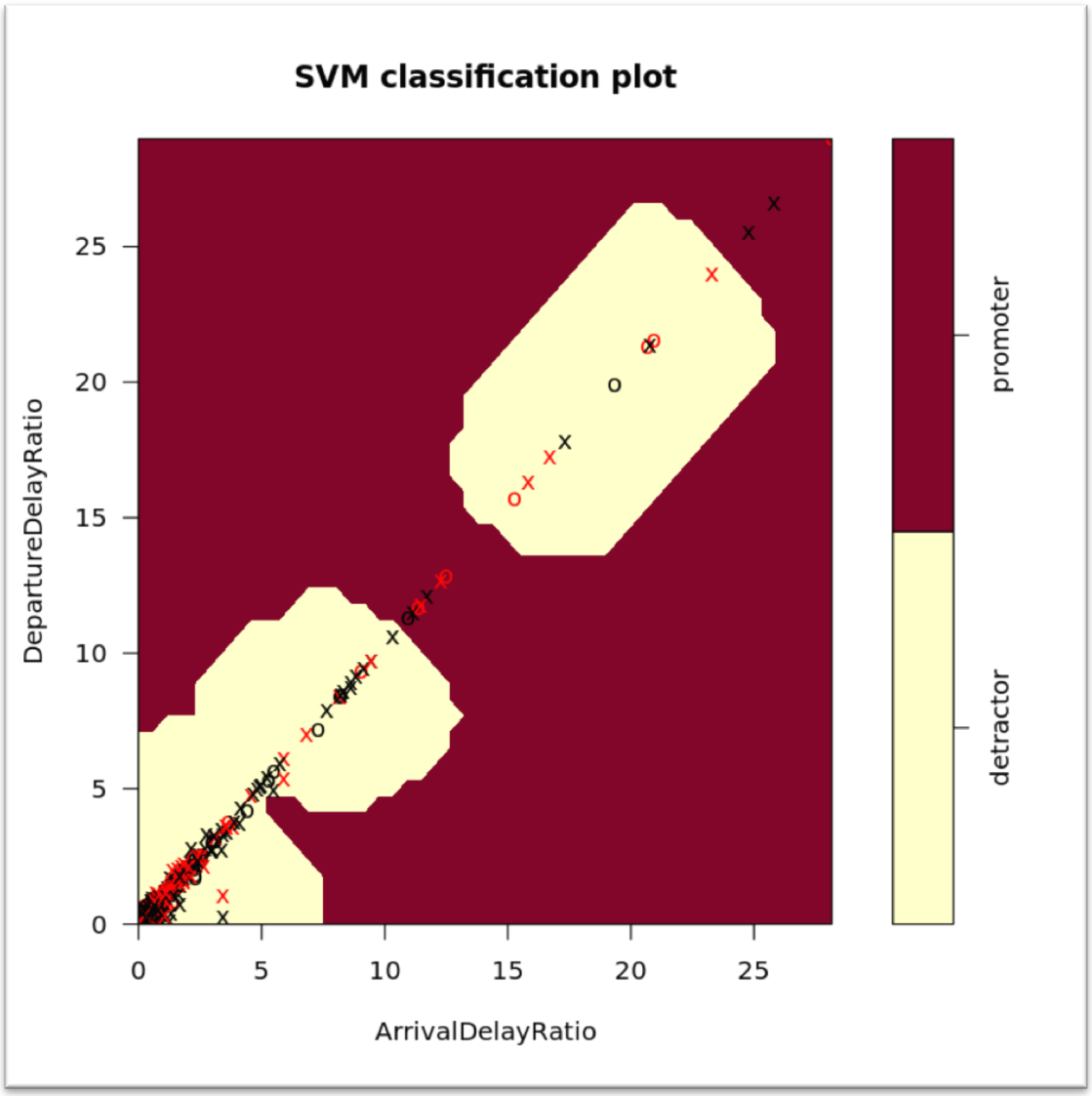
***Insight**: These graphs clearly present that when a passenger is more loyal or eating more at the airport, he/she is less likely to become a detractor. As the arrival delay ratio grows, a passenger is more likely to become a detractor. It is also worth noting that departure delay ratio affects differently on female and male passengers: when the arrival delay ratio is less than 15% (delay time is less than 15% of the flight time), female passengers are more likely to become detractors than male passengers; when the arrival delay ratio is more than 15%, male passengers become more likely to become detractors.*

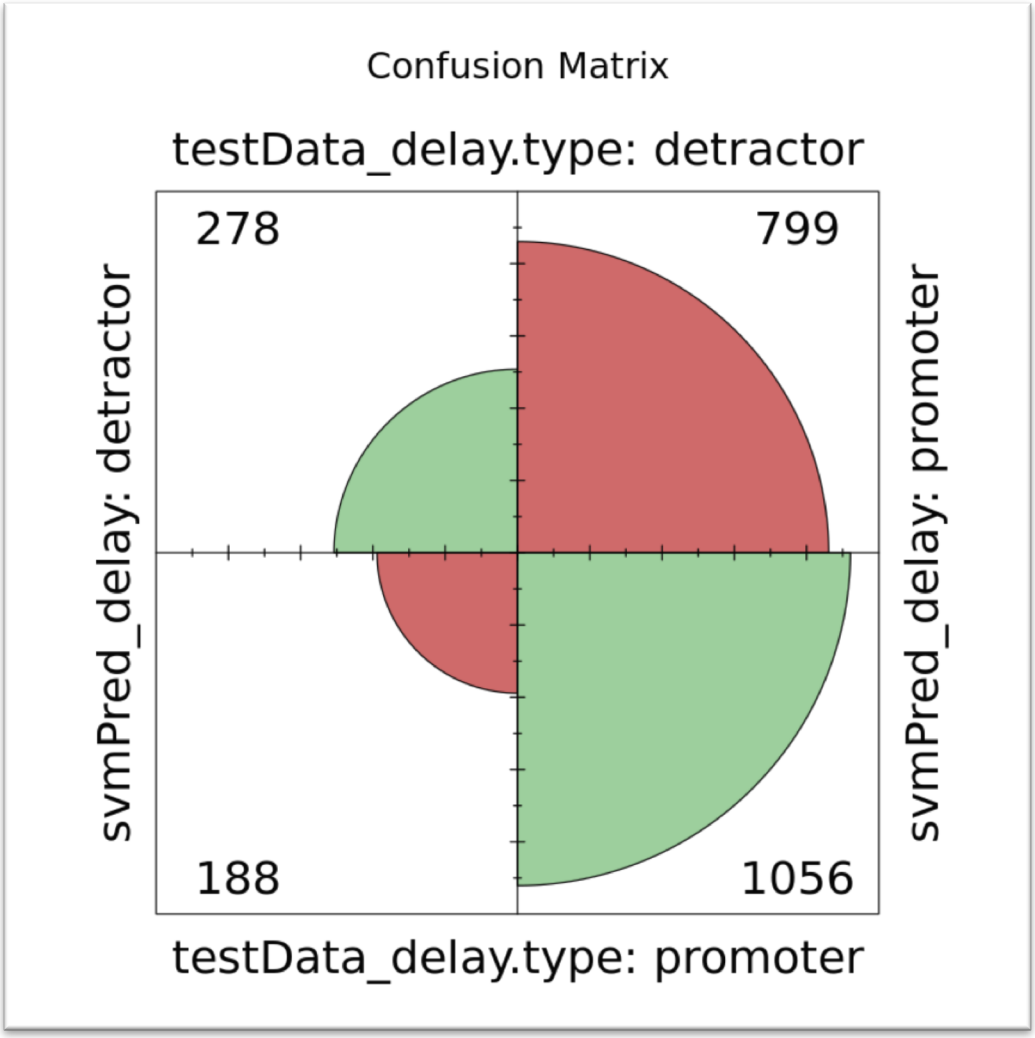
6.3. SVM

6.3.1. Model 1:

While creating this model, we were quite confident that the ArrivalDelayRatio and DepartureDelayRatio which we created were enough to get a satisfactory SVM Model test set accuracy.

We removed the data containing the passive type of reviewers and then we divided the dataset in 1/3rd for the test set and 2/3rd for the train set. After this, we ran the SVM model on the train set. We used the SVM model to predict the test set. Following are the results which we observed:


SVM classification plot
DepartureDelayRatio
ArrivalDelayRatio
0
5
10
15
20
25
promoter
detractor



***Insight**: We saw that these two attributes were not sufficient to correctly classify people as detractors or promoters. Therefore, we came to know there is something beyond flight cancellations.*

Model Accuracy on test Data: ***57.47%***

Code:

```
# Data frame subset
df_delay <- df[,c('DepartureDelayRatio','ArrivalDelayRatio','type')]
# Filtering passive passengers
df_delay <- df_delay %>%
  filter(type != 'passive')
# making type as factor
df_delay$type <- as.factor(as.character(df_delay$type))
randIndex <- sample(1:dim(df_delay)[1])
# Getting 2/3rd elements number count of the dataset
cutPoint2_3 <- floor(2 * dim(df_delay)[1]/3)
# Creating Training data based on two-thirds of the df_delay dataset
```

```
trainData_delay <- df_delay[randIndex[1:cutPoint2_3],]

# Creating Test data based on remaining one-third of the df_delay dataset
testData_delay <- df_delay[randIndex[(cutPoint2_3 +1):dim(df_delay)[1]],]
classifier_delay = svm(formula = type~.,
            data = trainData_delay,
            type = 'C-classification',
            kernel = "radial", gamma = 1, cost = 3)
plot(classifier_delay, data = testData_delay, DepartureDelayRatio~ArrivalDelayRatio)
svmPred_delay <- predict(classifier_delay, testData_delay)
compTable_delay <- data.frame(testData_delay$type, svmPred_delay)
compTable_delay <- table(compTable_delay)
100 - (compTable_delay["detractor","promoter"] +
compTable_delay["promoter","detractor"]) * 100 / (compTable_delay[1] +
compTable_delay[2] + compTable_delay[3] + compTable_delay[4])
fourfoldplot(compTable_delay, color = c("#CC6666", "#99CC99"),
        conf.level = 0, margin = 1, main = "Confusion Matrix")
```

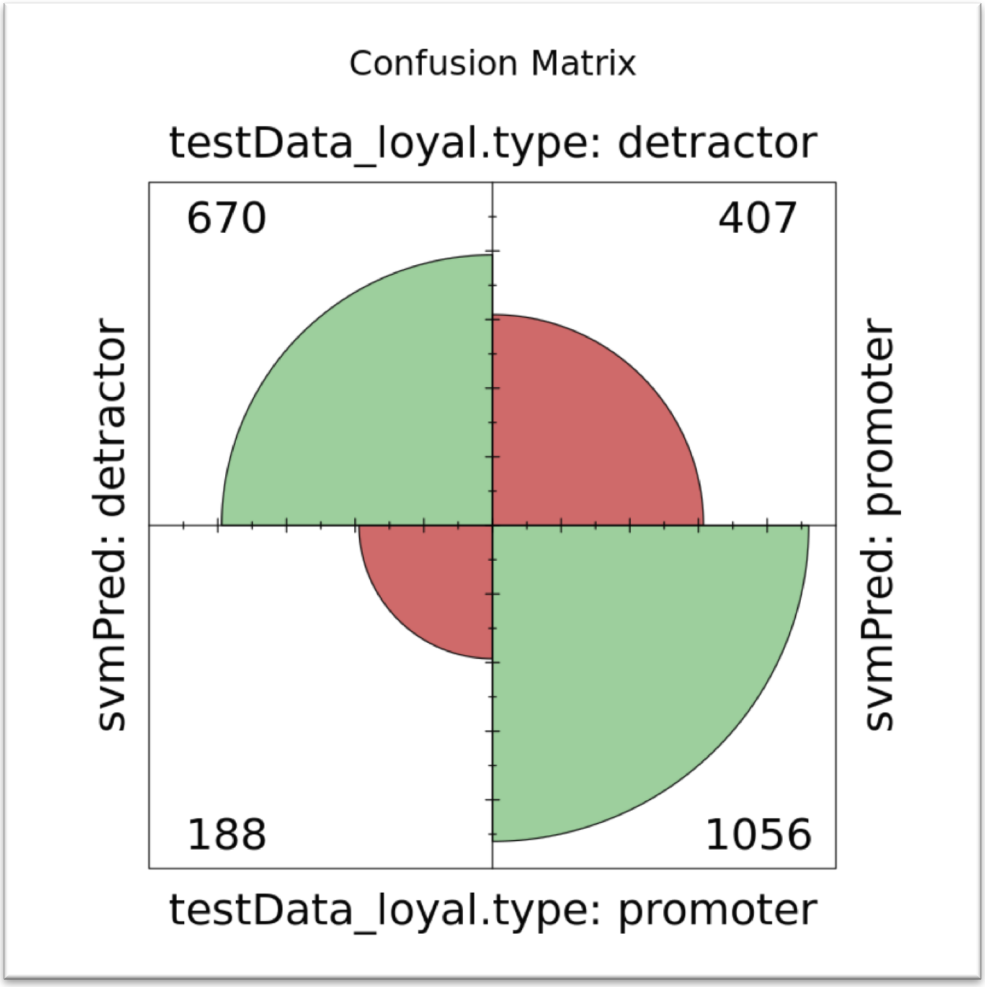
6.3.2. Model 2:

We wanted to verify that there was something beyond flight cancellations and therefore, we used several other attributes along with ArrivalDelayRatio and DepartureDelayRatio to test the model. They are as follows:

- Airline.Status
- Loyalty
- Total.Freq.Flyer.Acct
- DepartureDelayRatio
- ArrivalDelayRatio
- Age
- Price.Sensitivity
- Flights.Per.Year
- Type (Promoters and Detractor)

To prepare data that would support the smooth running of the SVM model, there was a need to convert categorical variables into continuous variables. Therefore, we converted Airline Status into a numeric variable by keeping Blue = 1, Silver = 2, Gold = 3, Platinum = 4.

We got the following output:



Model Accuracy on test Data: ***74.36%.***

But the problem with this kind of model is it doesn't really tell us what is going wrong for detractors and what is going right for promoters. Therefore, we had to use a different strategy.

Code:

```
df_loyal <- df[,c('Airline.Status',
                  'Loyalty',
                  'Total.Freq.Flyer.Accts',
                  'DepartureDelayRatio',
                  'ArrivalDelayRatio',
                  'Age',
                  'Price.Sensitivity',
```

```
                'Flights.Per.Year',
                'type')]

df_loyal$Total.Freq.Flyer.Accts <- as.numeric(df_loyal$Total.Freq.Flyer.Accts)

for(i in 1:nrow(df_loyal))
{
  if(df_loyal$Airline.Status[i] == 'Blue')
  {
    df_loyal$Airline.Status[i] <- 1
  }
  else if(df_loyal$Airline.Status[i] == 'Silver')
  {
    df_loyal$Airline.Status[i] <- 2
  }
  else if(df_loyal$Airline.Status[i] == 'Gold')
  {
    df_loyal$Airline.Status[i] <- 3
  }
  else if(df_loyal$Airline.Status[i] == 'Platinum')
  {
    df_loyal$Airline.Status[i] <- 4
  }
}

df_loyal$Airline.Status <- as.numeric(df_loyal$Airline.Status)

df_loyal <- df_loyal %>%
  filter(type != 'passive')

df_loyal$type <- as.factor(as.character(df_loyal$type))

# Creating Training data based on two-thirds of the df_loyal dataset
trainData_loyal <- df_loyal[randIndex[1:cutPoint2_3],]

# Creating Test data based on remaining one-third of the df_loyal dataset
testData_loyal <- df_loyal[randIndex[(cutPoint2_3 +1):dim(df_loyal)[1]],]

classifier_loyal = svm(formula = type~.,
                       data = trainData_loyal,
                      type = 'C-classification',
                       kernel = "radial", gamma = 0.05, cost = 2)
svmPred <- predict(classifier_loyal, testData_loyal)
# Confusion Matrix
compTable_df <- data.frame(testData_loyal$type, svmPred)
compTable <- table(compTable_df)
# Confusion Matrix plot
fourfoldplot(compTable, color = c("#CC6666", "#99CC99"),
          conf.level = 0, margin = 1, main = "Confusion Matrix")
```

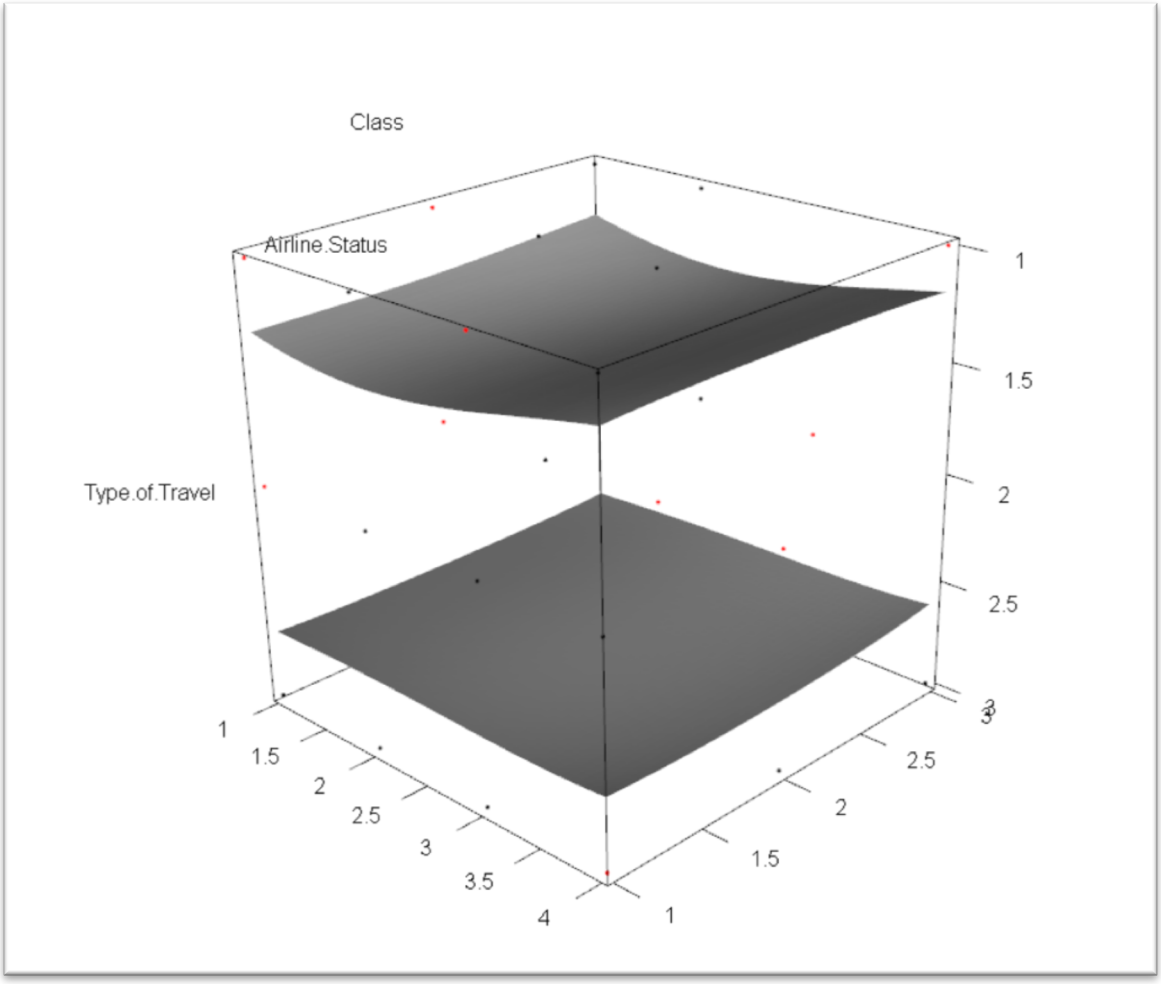
*100 - (compTable["detractor","promoter"] + compTable["promoter","detractor"]) * 100 / (compTable[1] + compTable[2] + compTable[3] + compTable[4])*

6.3.3. Model 3:

After the second model, we had to narrow down on some of the variables which we think would be useful. We took a higher level approach on deciding the variables. We thought what are the airline specific attributes that are associated with a passenger when he flies. What are the options he chooses before flying. Then we came down to three attributes:

- Airline.Status
- Type.of.Travel
- Class

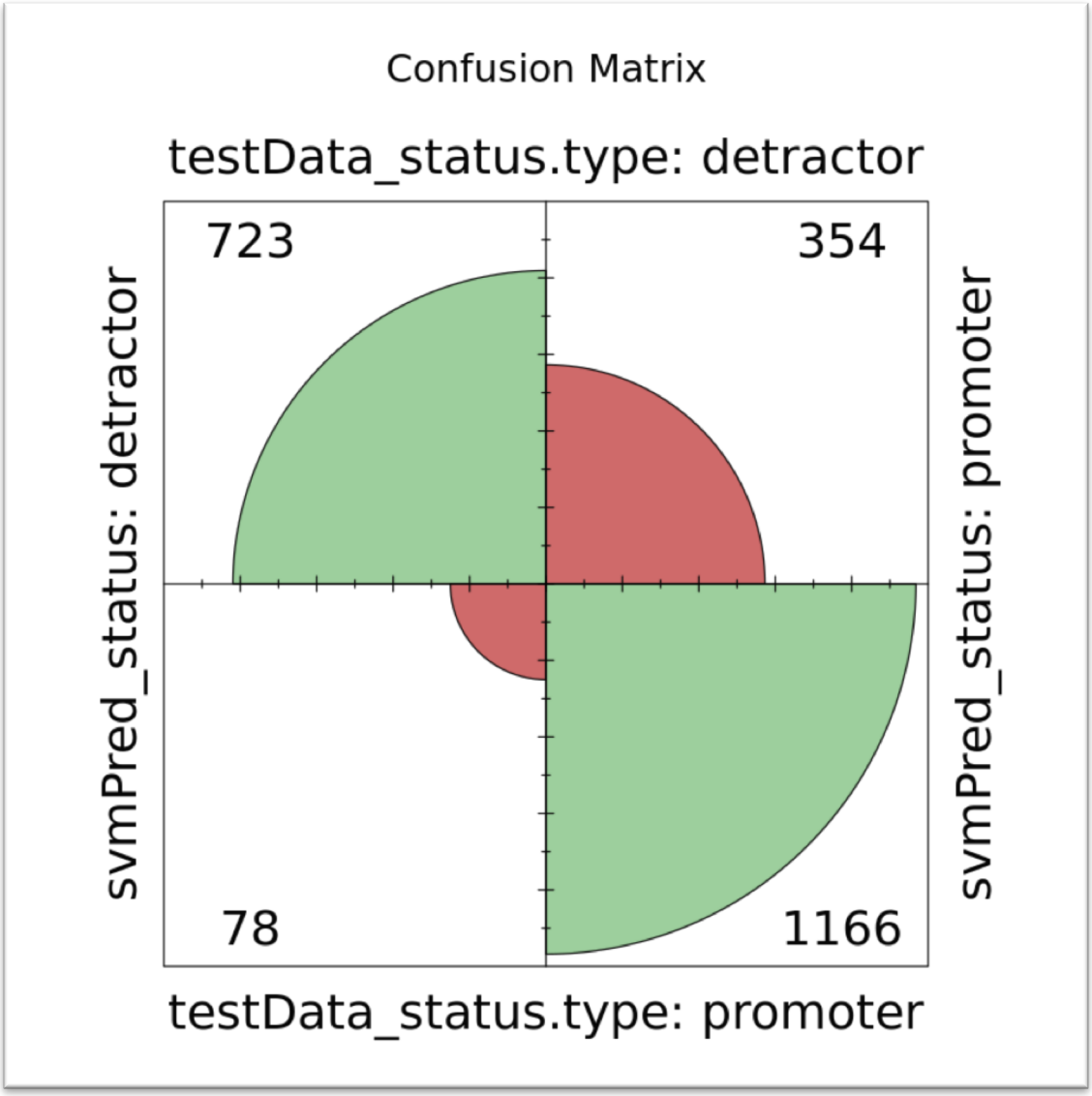
We decided to run the SVM model on these attributes. For these we had to convert them into continuous variables similar to Model 2. As we knew we were using 3 variables to predict promoters and detractors we could go for a better visualization technique than a normal plot function for 2D plot. Therefore, we created a free-flowing 3D visualization where a hyperplane separates the detractors from promoters.
We got the following output:



***Insight**: Blue dots on the plot represent promoters and red represent detractors. This output was much more insightful in terms of identifying what exactly is going wrong with detractors.*

******Model Accuracy on test Data: 81.38%***

From this, we could conclude that Airline Status, Type of Travel, and Class are the prominent indicators for predicting promoters and detractors.



Code:

```
df_status <- df[,c('Airline.Status',
                   'Type.of.Travel',
                   'Class',
                  'type')]

for(i in 1:nrow(df_status))
{
  if(df_status$Airline.Status[i] == 'Blue')
  {
    df_status$Airline.Status[i] <- 1
  }
  else if(df_status$Airline.Status[i] == 'Silver')
  {
    df_status$Airline.Status[i] <- 2
  }
  else if(df_status$Airline.Status[i] == 'Gold')
```

```
  {
    df_status$Airline.Status[i] <- 3
  }
  else if(df_status$Airline.Status[i] == 'Platinum')
  {
    df_status$Airline.Status[i] <- 4
  }
}

df_status$Airline.Status <- as.numeric(df_status$Airline.Status)

for(i in 1:nrow(df_status))
{
  if(df_status$Class[i] == 'Eco')
  {
    df_status$Class[i] <- 1
  }
  else if(df_status$Class[i] == 'Eco Plus')
  {
    df_status$Class[i] <- 2
  }
  else if(df_status$Class[i] == 'Business')
  {
    df_status$Class[i] <- 3
  }
}

df_status$Class <- as.numeric(df_status$Class)

for(i in 1:nrow(df_status))
{
  if(df_status$Type.of.Travel[i] == 'Mileage tickets')
  {
    df_status$Type.of.Travel[i] <- 1
  }
  else if(df_status$Type.of.Travel[i] == 'Personal Travel')
  {
    df_status$Type.of.Travel[i] <- 2
  }
  else if(df_status$Type.of.Travel[i] == 'Business travel')
  {
    df_status$Type.of.Travel[i] <- 3
  }
}

df_status$Type.of.Travel <- as.numeric(df_status$Type.of.Travel)

df_status <- df_status %>%
  filter(type != 'passive')
```

```
df_status$type <- as.factor(as.character(df_status$type))

# Creating Training data based on two-thirds of the df_loyal dataset
trainData_status <- df_status[randIndex[1:cutPoint2_3],]

# Creating Test data based on remaining one-third of the df_loyal dataset
testData_status <- df_status[randIndex[(cutPoint2_3 +1):dim(df_status)[1]],]

classifier_status = svm(formula = type~.,
                        data = trainData_status,
                       type = 'C-classification',
                        kernel = "radial", gamma = 0.05, cost = 2)
svmPred_status <- predict(classifier_status, testData_status)
# Confusion Matrix
compTable_status <- data.frame(testData_status$type, svmPred_status)
compTable_status <- table(compTable_status)
# Confusion Matrix plot
fourfoldplot(compTable_status, color = c("#CC6666", "#99CC99"),
          conf.level = 0, margin = 1, main = "Confusion Matrix")
error_rate              <-              (compTable_status["detractor","promoter"]              +
compTable_status["promoter","detractor"])    *    100    /    (compTable_status[1]    +
compTable_status[2] + compTable_status[3] + compTable_status[4])
# 3D plot
trainData_status$type <- ifelse(trainData_status[,4] == 'promoter', 1, 2 )

# Actual plot
plot3d(trainData_status[,-4], col = trainData_status$type)

nnew = 50
newdat.list = lapply(trainData_status[,-4], function(x) seq(min(x), max(x), len=nnew))
newdat      = expand.grid(newdat.list)
newdat.pred = predict(classifier_status, newdata=newdat, decision.values=T)
newdat.dv   = attr(newdat.pred, 'decision.values')
newdat.dv   = array(newdat.dv, dim=rep(nnew, 3))

contour3d(newdat.dv,                level=0,                x=newdat.list$Airline.Status,
y=newdat.list$Type.of.Travel, z=newdat.list$Class, add=T)
```

6.4. ASSOCIATION RULE MINING

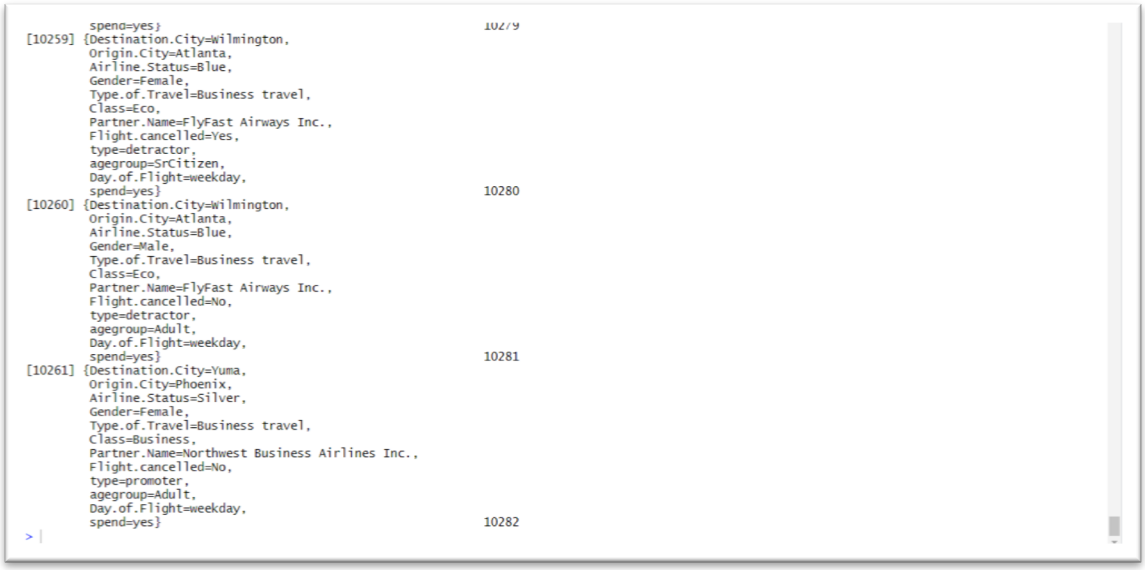
We used association rule mining to understand what are the major factors affecting the likelihood to recommend. We wanted to know what combination of attributes are leading towards promoters and detractors as well. Further, we related our findings to airlines to get more deeper insight.

To start with, we created a subset of all the categorical variables. After converting it into transaction matrix, we ran analysis on it which resulted into many rules.

CODE :

```
library(arules)
library(arulesViz)
subset_df <- df[c('Destination.City', 'Origin.City','Airline.Status', 'Gender', 'Type.of.Travel', 'Class', 'Partner.Name', 'Flight.cancelled', 'type', 'agegroup', 'Day.of.Flight', 'spend')]
subsetX_df<- as(subset_df, "transactions")
arules::inspect(subsetX_df)
```

OUTPUT :

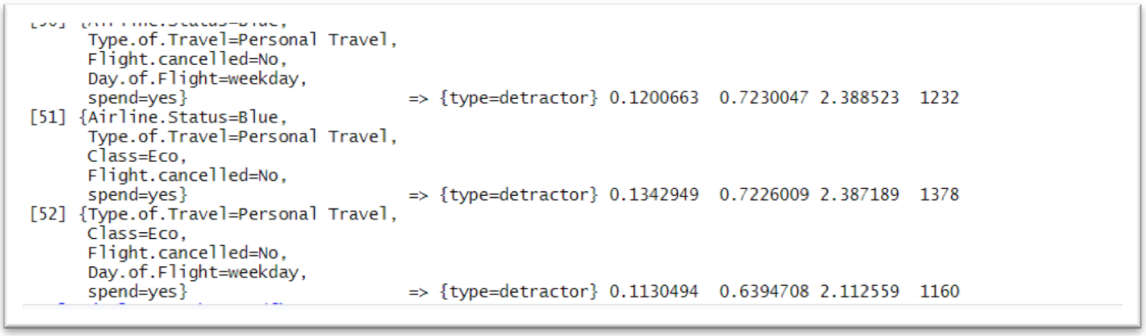


Insight: *More than ten thousand rules were generated which were difficult to interpret. Hence we just focused on Detractors and generated rules using apriori function.*

CODE :

```
ruleset_subsetX_df<- apriori(subsetX_df,
                parameter= list(support= 0.1, confidence = 0.5),
                appearance = list(default = "lhs", rhs=("type=detractor")))
arules::inspect(ruleset_subsetX_df)
```

OUTPUT :



```
      Type.of.Travel=Personal Travel,
      Flight.cancelled=No,
      Day.of.Flight=weekday,
      spend=yes}                    => {type=detractor} 0.1200663  0.7230047 2.388523  1232
[51] {Airline.Status=Blue,
      Type.of.Travel=Personal Travel,
      Class=Eco,
      Flight.cancelled=No,
      spend=yes}                    => {type=detractor} 0.1342949  0.7226009 2.387189  1378
[52] {Type.of.Travel=Personal Travel,
      Class=Eco,
      Flight.cancelled=No,
      Day.of.Flight=weekday,
      spend=yes}                    => {type=detractor} 0.1130494  0.6394708 2.112559  1160
```

Insight: Here we got an overview of all the combinations of attributes that are causing the customer to become a detractor. Then using this information we started making different combinations, for both promoters and detractors, and observed the rules generated from them. We kept the parameter lift as high as possible and plotted the rules for better visualization.

6.4.1. AIRLINE STATUS | TYPE OF TRAVEL | CLASS

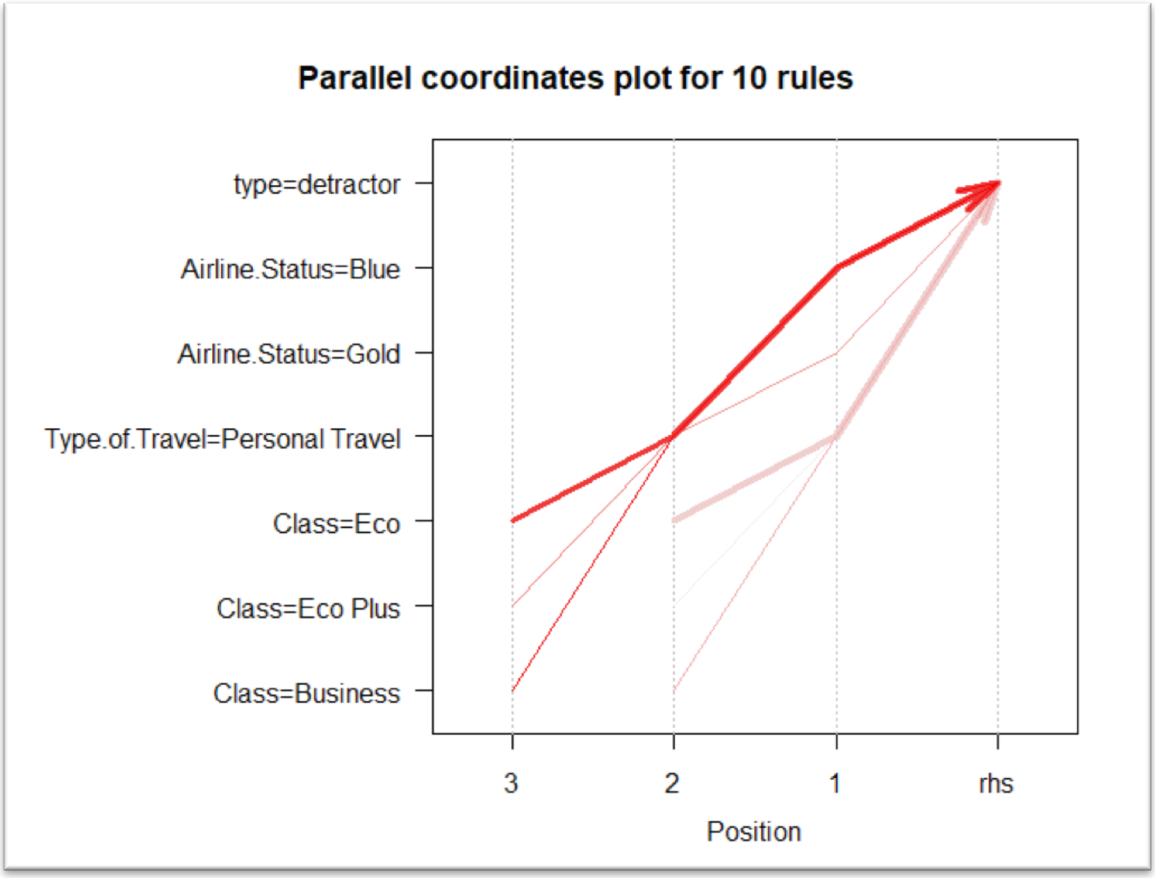
CODE:

```
 subset1_df<- df[c('Airline.Status','Type.of.Travel', 'Class','type')]
subset1X_df<- as(subset1_df, "transactions")
```

FOR DETRACTORS

CODE :

```
ruleset1_subsetX_df<- apriori(subset1X_df,
                    parameter= list(support= 0.01, confidence = 0.05),
                    appearance = list(default = "lhs", rhs=("type=detractor")))
arules::inspect(ruleset1_subsetX_df)
goodRules<- ruleset1_subsetX_df[quality(ruleset1_subsetX_df)$lift>2]
arules::inspect(goodRules)

# PLOT #
d_AirlineStatus_TypeOfTravel_Class<-    plot(goodRules,    method="paracoord",
control=list(reorder=TRUE))
```

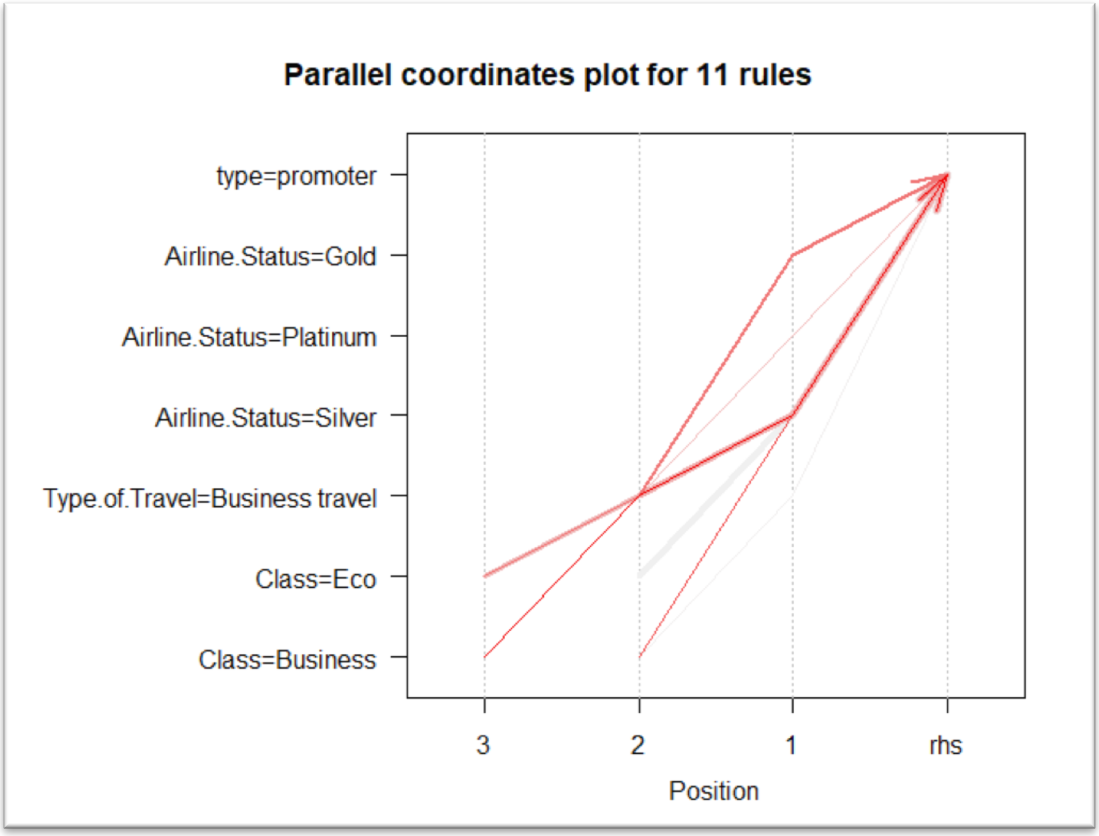


Insight: Here we understand that customers with eco class, type of travel personal and airline status blue are not satisfied with their service.

FOR PROMOTERS

CODE *:*

```
ruleset2_subsetX_df<- apriori(subset1X_df,
                      parameter= list(support= 0.01, confidence = 0.05),
                      appearance = list(default = "lhs", rhs=("type=promoter")))
arules::inspect(ruleset2_subsetX_df)
goodRules2<- ruleset2_subsetX_df[quality(ruleset2_subsetX_df)$lift>1.3]
arules::inspect(goodRules2)
#PLOT
p_AirlineStatus_TypeOfTravel_Class<-  plot(goodRules2,  method="paracoord",
control=list(reorder=TRUE))
```



Insight: Hence, customers giving high score of likelihood to recommend are from business class, with airline status silver and type of travel business.

6.4.2. PARTNER NAME | GENDER | AGEGROUP

CODE :

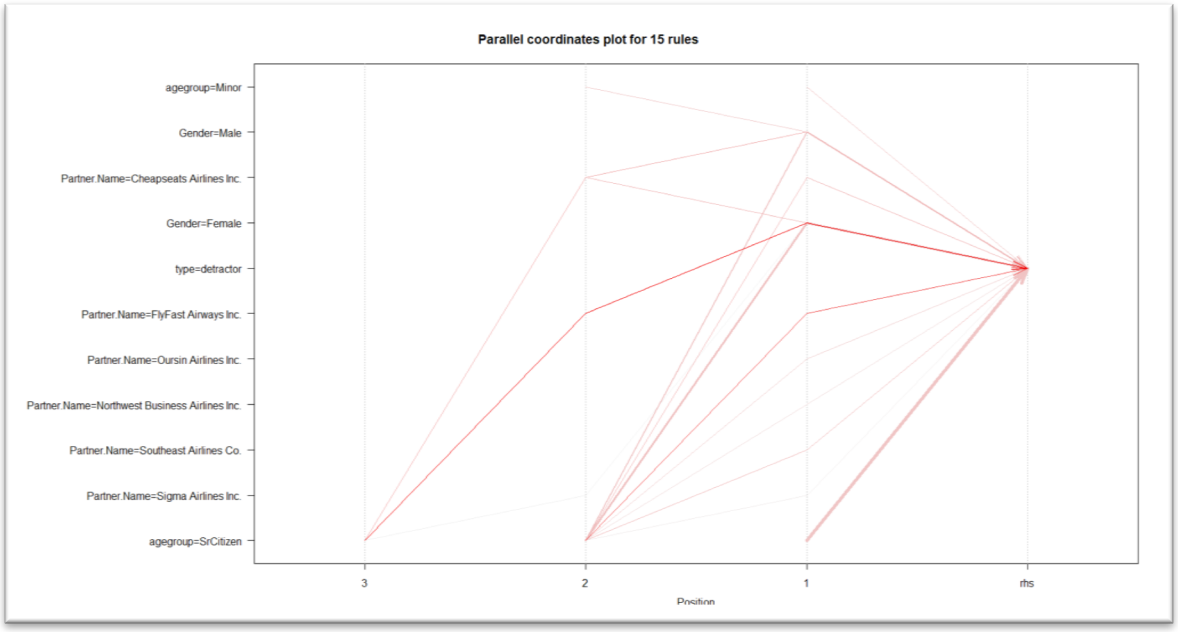
```
subset2_df<- df[c('Gender','Partner.Name', 'agegroup','type')]
subset2X_df<- as(subset2_df, "transactions")
```

FOR DETRACTORS

CODE :

```
ruleset3_subsetX_df<- apriori(subset2X_df,
                    parameter= list(support= 0.01, confidence = 0.05),
                    appearance = list(default = "lhs", rhs=("type=detractor")))

arules::inspect(ruleset3_subsetX_df)

goodRules3<- ruleset3_subsetX_df[quality(ruleset3_subsetX_df)$lift>1.5]
arules::inspect(goodRules3)

# PLOT #
```

plot(goodRules3, method="paracoord", control=list(reorder=TRUE))

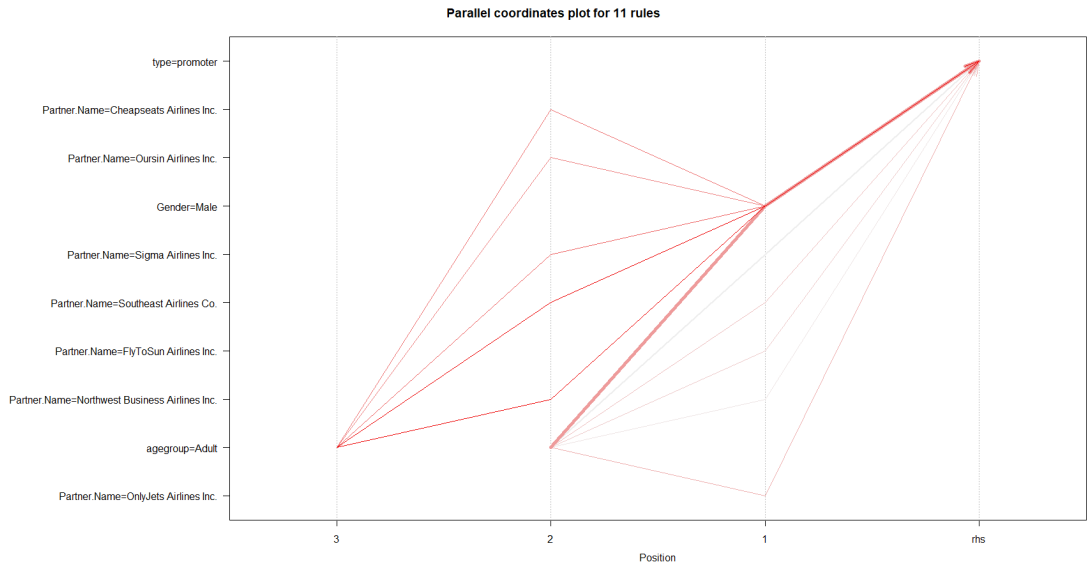


Insight: We observed that female travellers who are senior citizen travelling with partner airline Flyfast Airways Inc. are detractors.

FOR PROMOTERS

CODE :

```
ruleset4_subsetX_df<- apriori(subset2X_df,
                      parameter= list(support= 0.01, confidence = 0.05),
                      appearance = list(default = "lhs", rhs=("type=promoter")))
arules::inspect(ruleset4_subsetX_df)
goodRules4<- ruleset4_subsetX_df[quality(ruleset4_subsetX_df)$lift>1.3]
arules::inspect(goodRules4)
# PLOT #
plot(goodRules4, method="paracoord", control=list(reorder=TRUE))
```



Insight: Whereas, adult male travellers are promoters. There are a few partner names highlighted but we still cannot see a prominent name. Hence, we took help of barplots to determine a promoter partner airline.

6.4.3. TYPE OF TRAVEL | FLIGHT CANCELLED | DAY OF FLIGHT

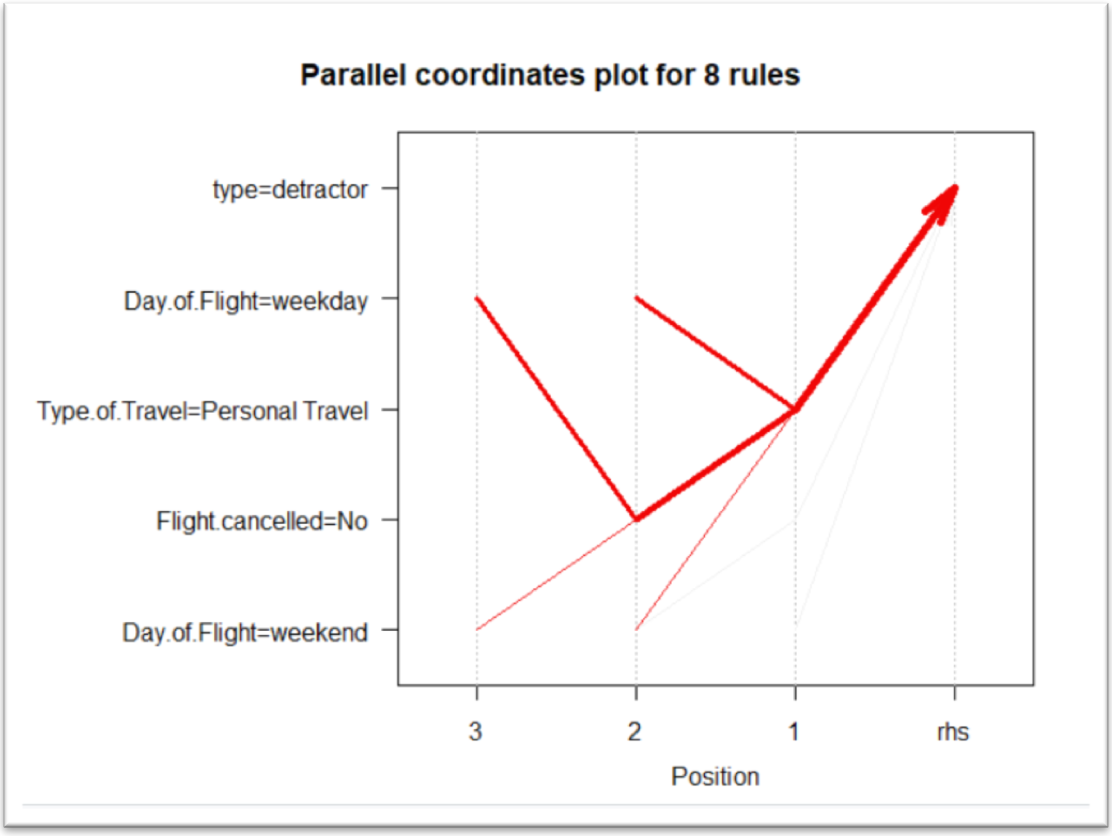
CODE *:*

```
subset3_df<- df[c('Type.of.Travel','Flight.cancelled', 'Day.of.Flight','type')]
subset3X_df<- as(subset3_df, "transactions")
```

FOR DETRACTORS

CODE :

```
ruleset5_subsetX_df<- apriori(subset3X_df,
                    parameter= list(support= 0.01, confidence = 0.05),
                    appearance = list(default = "lhs", rhs=("type=detractor")))
arules::inspect(ruleset5_subsetX_df)
goodRules5<- ruleset5_subsetX_df[quality(ruleset5_subsetX_df)$lift>1.0]
arules::inspect(goodRules5)
# PLOT #
plot(goodRules5, method="paracoord", control=list(reorder=TRUE))
```

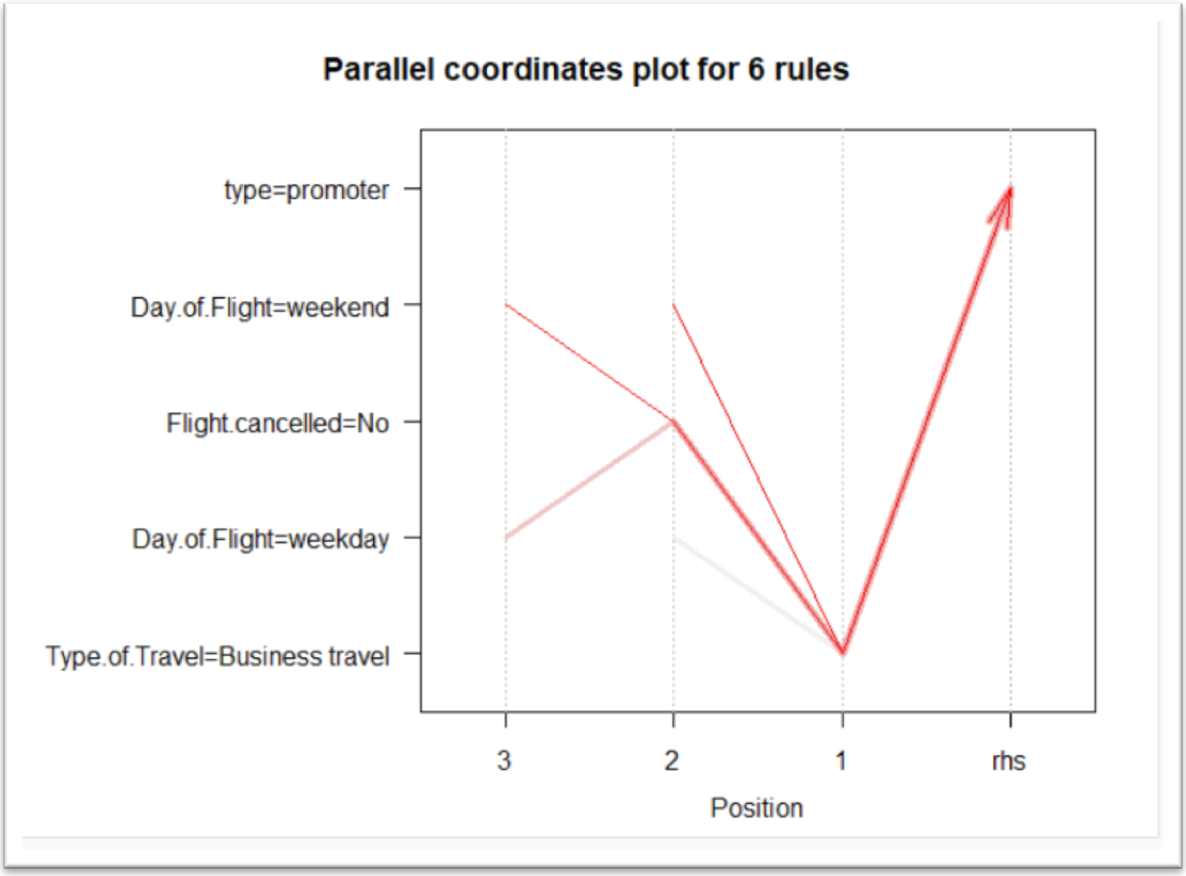


Insight: Here we see that flights on weekdays are detractors with type of travel personal.

FOR PROMOTERS

CODE :

```
ruleset6_subsetX_df<- apriori(subset3X_df, parameter= list(support= 0.01, confidence =0.05), appearance = list(default = "lhs", rhs=("type=promoter")))
arules::inspect(ruleset6_subsetX_df)
goodRules6<- ruleset6_subsetX_df[quality(ruleset6_subsetX_df)$lift>1.3]
arules::inspect(goodRules6)
```



Insight: Whereas people travelling for business, taking weekend flights are likely to become promoters.

6.4.4. TYPE OF TRAVEL | FLIGHT CANCELLED | CLASS

CODE *:*

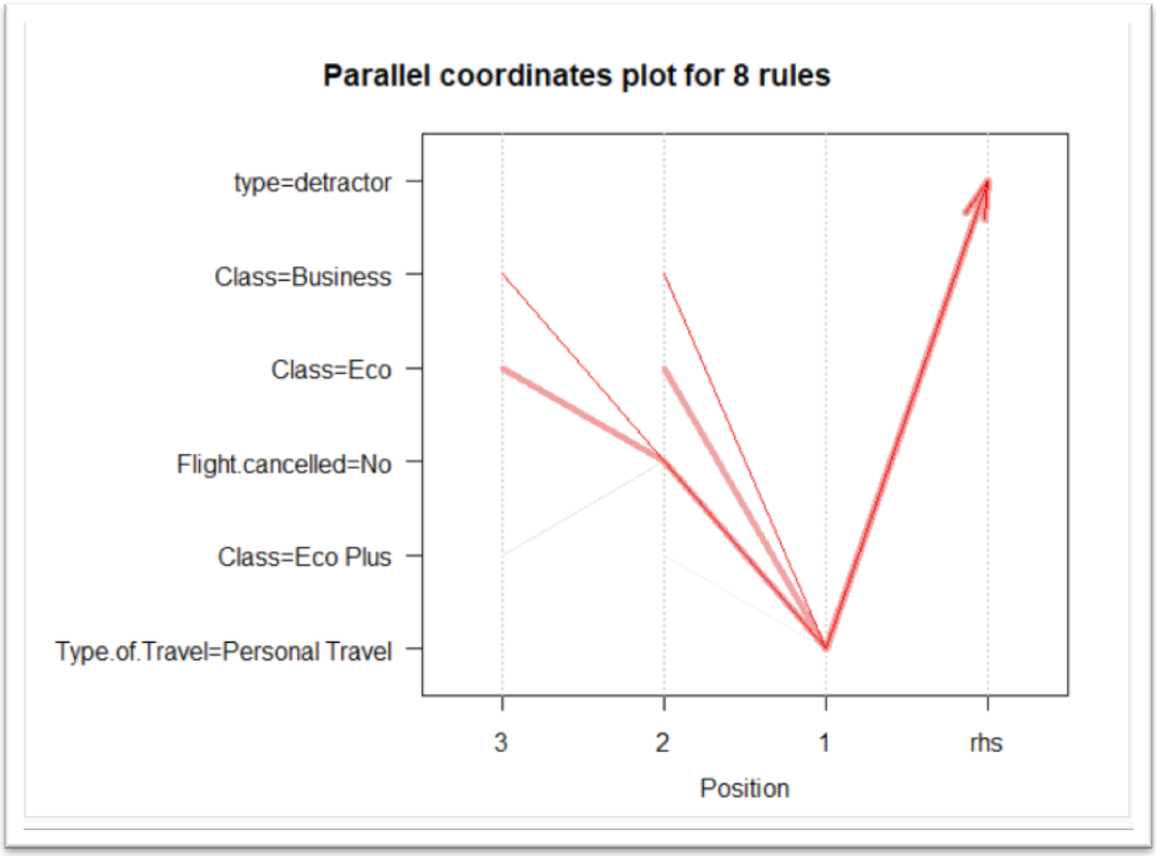
```
subset4_df<- df[c('Type.of.Travel','Flight.cancelled', 'Class','type')]
subset4X_df<- as(subset4_df, "transactions")
```

FOR DETRACTORS

CODE :

```
ruleset7_subsetX_df<- apriori(subset4X_df,
                      parameter= list(support= 0.01, confidence = 0.05),
                      appearance = list(default = "lhs", rhs=("type=detractor")))
arules::inspect(ruleset7_subsetX_df)
goodRules7<- ruleset7_subsetX_df[quality(ruleset7_subsetX_df)$lift>1.5]
arules::inspect(goodRules7)

# PLOT #
```

plot(goodRules7, method="paracoord", control=list(reorder=TRUE))

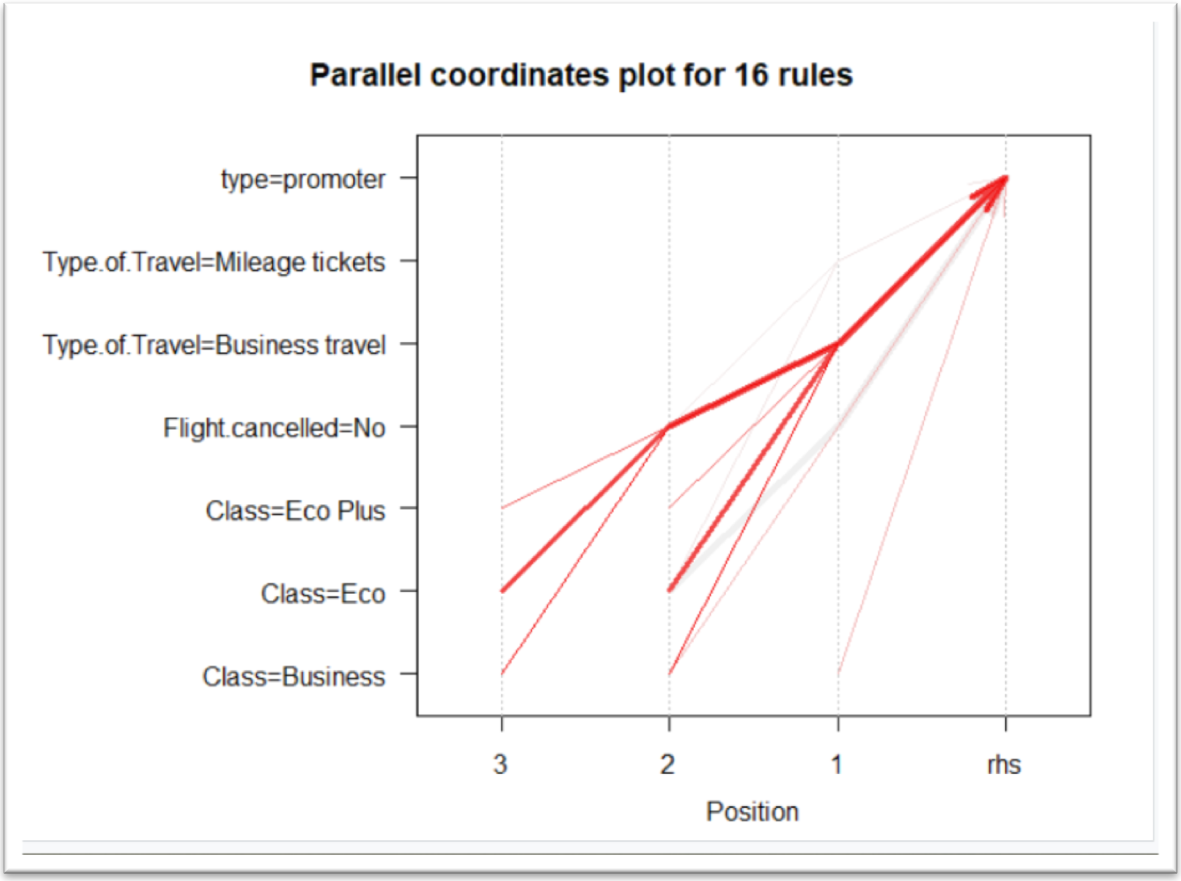


Insight: Customers with type of travel personal, travelling in eco class are giving low recommendation scores.

FOR PROMOTERS

CODE :

```
ruleset8_subsetX_df<- apriori(subset4X_df,
                      parameter= list(support= 0.01, confidence = 0.05),
                      appearance = list(default = "lhs", rhs=("type=promoter")))
arules::inspect(ruleset8_subsetX_df)
goodRules8<- ruleset8_subsetX_df[quality(ruleset8_subsetX_df)$lift>1.0]
arules::inspect(goodRules8)
# PLOT #
plot(goodRules8, method="paracoord", control=list(reorder=TRUE))
```



Insight: And with the class being same, that is eco, customers travelling for business are promoters.

6.4.5. TYPE OF TRAVEL | DAY OF FLIGHT | CLASS

CODE *:*

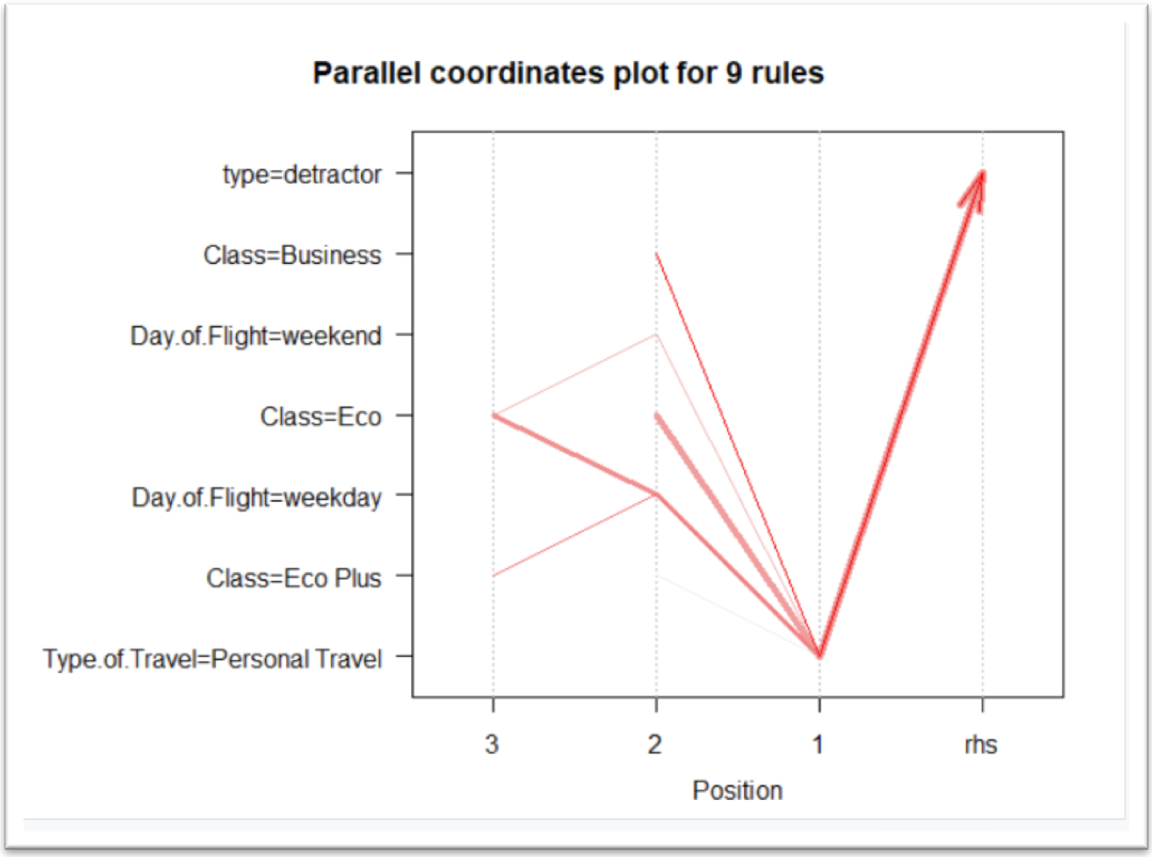
```
subset5_df<- df[c('Type.of.Travel','Day.of.Flight', 'Class','type')]
subset5X_df<- as(subset5_df, "transactions")
```

FOR DETRACTORS

CODE *:*

```
ruleset9_subsetX_df<- apriori(subset5X_df,
                      parameter= list(support= 0.01, confidence = 0.05),
                      appearance = list(default = "lhs", rhs=("type=detractor")))
arules::inspect(ruleset9_subsetX_df)
goodRules9<- ruleset9_subsetX_df[quality(ruleset9_subsetX_df)$lift>1.5]
arules::inspect(goodRules9)
```

 # PLOT #
plot(goodRules9, method="paracoord", control=list(reorder=TRUE))

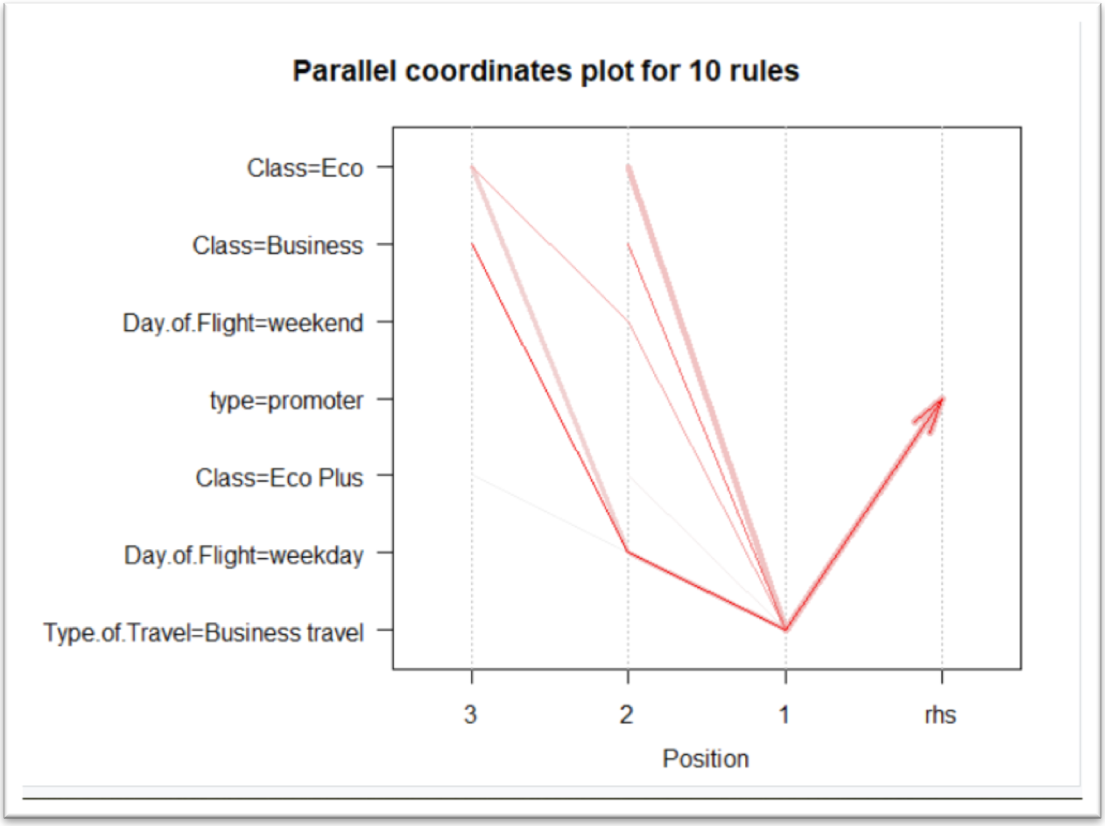


Insight: Lastly, in this combination it is again verified that customers travelling with class eco on a weekday with type of travel personal are very likely to become detractors.

FOR PROMOTERS

CODE :

```
ruleset10_subsetX_df<- apriori(subset5X_df,
                        parameter= list(support= 0.01, confidence = 0.05),
                        appearance = list(default = "lhs", rhs=("type=promoter")))
arules::inspect(ruleset10_subsetX_df)
goodRules10<- ruleset10_subsetX_df[quality(ruleset10_subsetX_df)$lift>1.3]
arules::inspect(goodRules10)
# PLOT #
plot(goodRules10, method="paracoord", control=list(reorder=TRUE))
```



Insight: But here we see for promoters, the type of travel is business as observed before, type customers are travelling on a weekday with business class.

To summarize, from all of these plots our analysis is, the strong indicators for detractors are customers travelling in class eco, airline status blue and type of travel personal and customers travelling in class business with airline status silver and type of travel business are promoters. We select these three variables out of all the others is because they are prominently observed giving constant result in all of the combinations that they were used.

Further, using the bar plot of partner name versus NPS, we had come to know that airline with lowest score was Flyfast Airways Inc. and the one with highest score was West Airways Inc. But we found out the count of data for each airline and observed that data for West Airways Inc. was too low to be compared with Flyfast Airways Inc.

CODE *:*
df%>% group_by(Partner.Name)%>% summarise(n=n())

OUTPUT :

```
 1 Cheapseats Airlines Inc.          2175
 2 Cool&Young Airlines Inc.           120
 3 EnjoyFlying Air Services           478
 4 FlyFast Airways Inc.              1197
 5 FlyHere Airways                    228
 6 FlyToSun Airlines Inc.             305
 7 GoingNorth Airlines Inc.           153
 8 Northwest Business Airlines Inc.  1219
 9 OnlyJets Airlines Inc.             356
10 Oursin Airlines Inc.               987
11 Paul Smith Airlines Inc.           537
12 Sigma Airlines Inc.               1595
13 Southeast Airlines Co.             897
14 West Airways Inc.                   14
```

Hence we considered the next best airline Northwest Business Airline Inc. ,with a significant number of data, as promoter.

Further, we wanted to know what are the factors affecting both of these airlines to a detractor or promoter respectively. So we started with comparison of average values of continuous variables, which interestingly resulted not much difference.

	Partner.Name	n	avg_age	avg_price_sensitivity	avg_loyalty	avg_flyer_accounts	avg_arr_delay_ratio	avg_dep_delay_ratio	avg_shopping	avg_eating
1	FlyFast Airways Inc.	1197	45.93400	1.284043	-0.2736936	0.8629908	0.9693697	0.9862471	26.28906	67.95906
2	Northwest Business Airlines Inc.	1219	45.94094	1.267588	-0.2531380	0.8564397	0.8840571	0.8718357	28.70960	67.97375

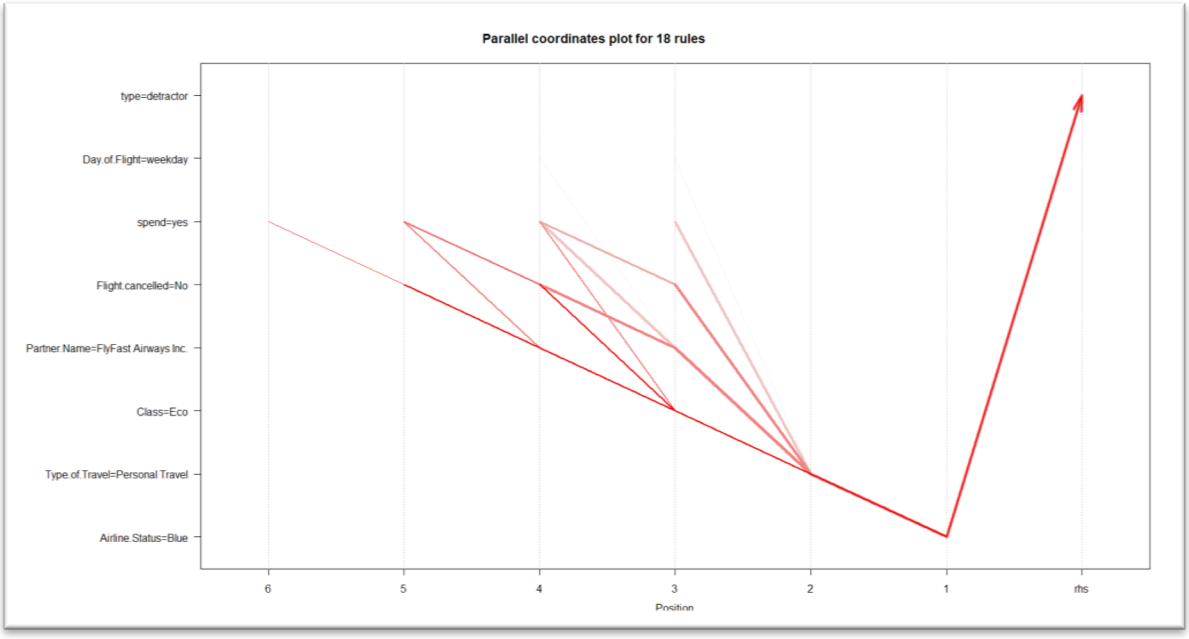
Hence, we considered the discrete variables selected above, class, airline status and type of travel and used association rule mining to understand the pattern and verify whether the above analysis applies for these airlines.

6.4.6. FLYFAST AIRWAYS INC.

CODE *:*

```
flyfast <- subset(subset_df, Partner.Name == "FlyFast Airways Inc.")
flyfastX<- as(flyfast, "transactions")
ruleset_flyfastX_df<- apriori(flyfastX,
                              parameter= list(support= 0.15, confidence = 0.8),
```

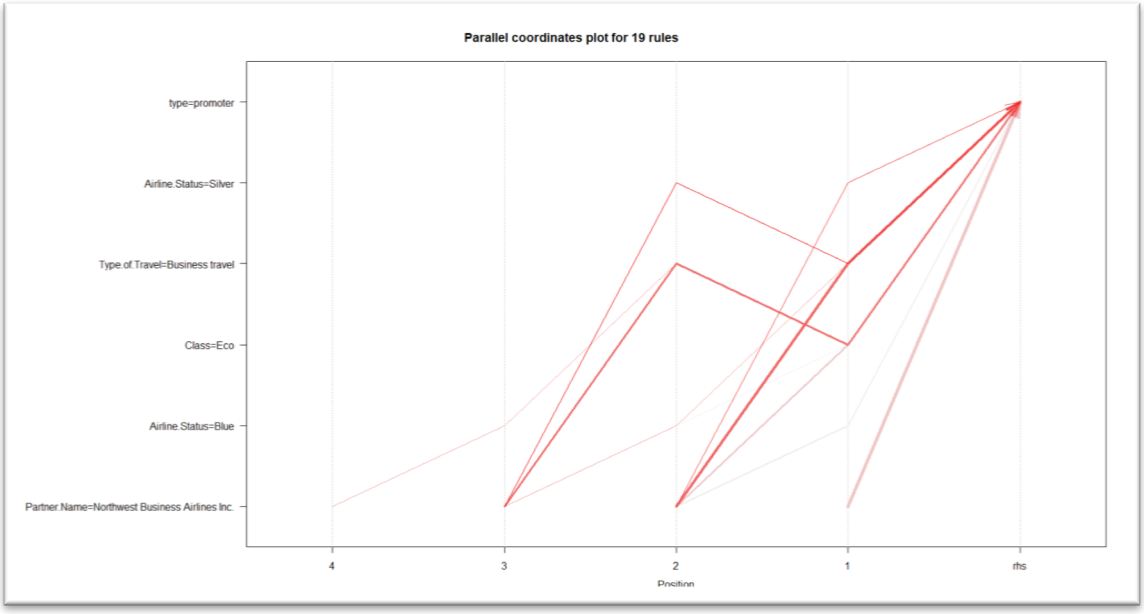
appearance = list(default = "lhs", rhs=("type=detractor")))
arules::inspect(ruleset_flyfastX_df)
flyfast_detractor<- plot(ruleset_flyfastX_df, method="paracoord", control=list(reorder=TRUE))



Insight: Here we can clearly see that customers travelling in Flyfast Airways with class eco, type of travel personal and airline status blue are detractors.

6.4.7. NORTHWESTERN BUSINESS AIRLINES INC.

CODE *:*
northwest<- subset(subset_df, Partner.Name == "Northwest Business Airlines Inc.")
View(northwest)
northwestX<- as(northwest, "transactions")
ruleset_northwestX_df<- apriori(northwestX,
parameter= list(support= 0.3, confidence = 0.55),
appearance = list(default = "lhs", rhs=("type=promoter")))
arules::inspect(ruleset_northwestX_df)
northwest_promoter<- plot(ruleset_subsetnorthwestX_df, method="paracoord", control=list(reorder=TRUE))



Insight: Whereas, the factors leading Northwestern Business Airlines to be a promoter are airline status silver and type of travel business.

6.5. MODEL COMPARISON

In this section, we compare the performance of the four models: linear regression, logistic regression, SVM, and association rule mining.

Model	Performance
Linear Regression	R square: 0.09
Logistic Regression	Accuracy: 0.82
SVM	Accuracy: 0.81
Association Rule Mining	Best for interpretation

After we try the models on all possible variables, we find that Class', 'Airline Status', and 'Type of Travel' are the most prominent predictor for customer type (detractor or promoter). Although both logistic regression and SVM achieve relatively high accuracy in predicting detractors, it is difficult to understand how each individual variable in the model contributes to the final outcome. In contrast, by presenting in what specific situation (blue

airline status, personal travel, and economy class) a passenger is the most likely to be a detractor, the association rule mining offers us clear and crucial business insights. Therefore, we think that the association rule mining is the best choice in this scenario.

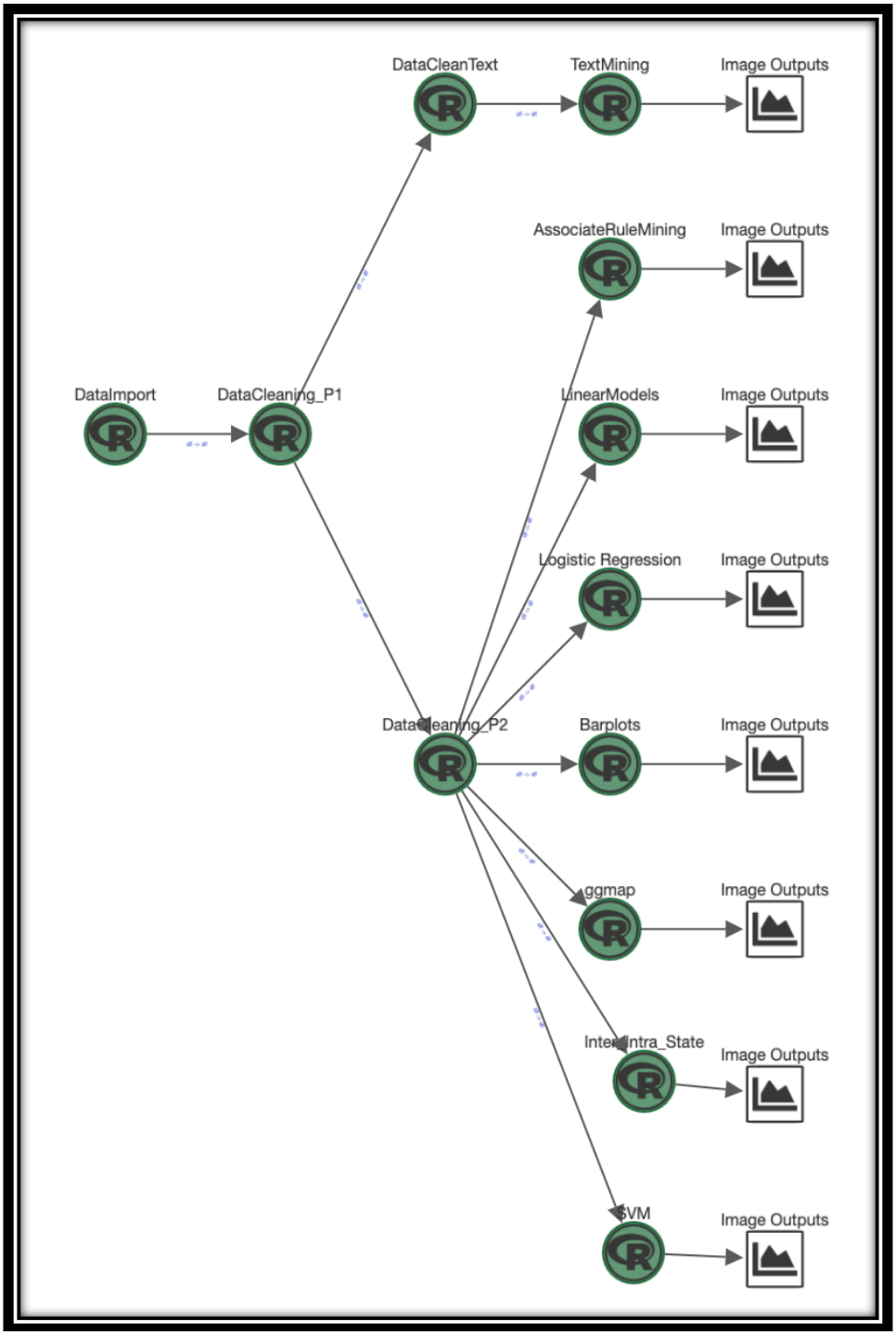
7. ACTIONABLE INSIGHTS

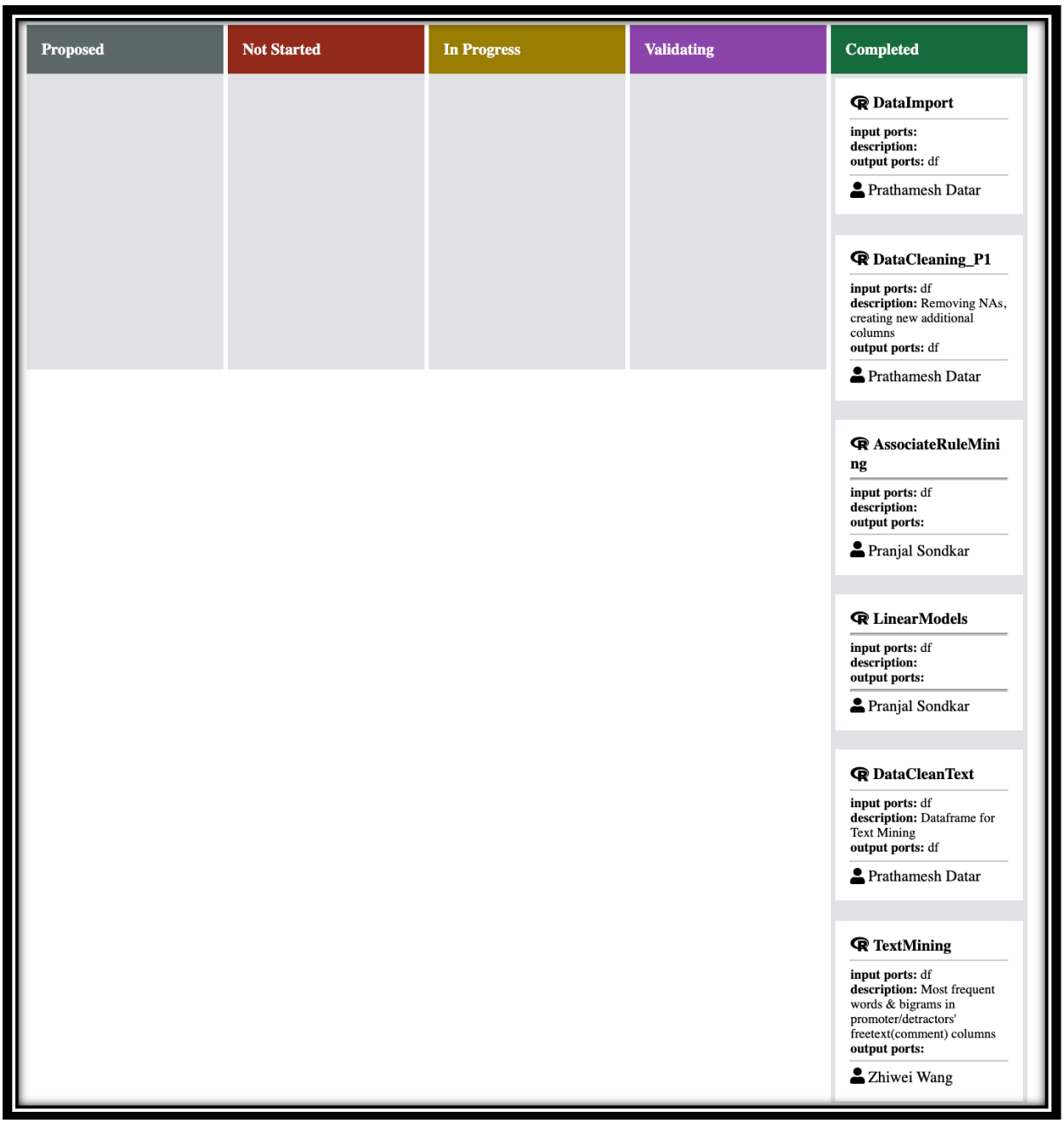
1. Improve Customer Service, such as luggage handling and leg room.
2. Work on Facilities and Services for Female, Minors, and Senior Citizens.
3. Improve Facilities and Services for Customers who are on Personal Travel, in Blue Airline Status, or in Eco-Class.
4. Further Investigation for the Airline Situation in Texas, especially Dallas.
5. Investigate the air routes with FlyFast as the Partner Airlines.
6. Comparison:

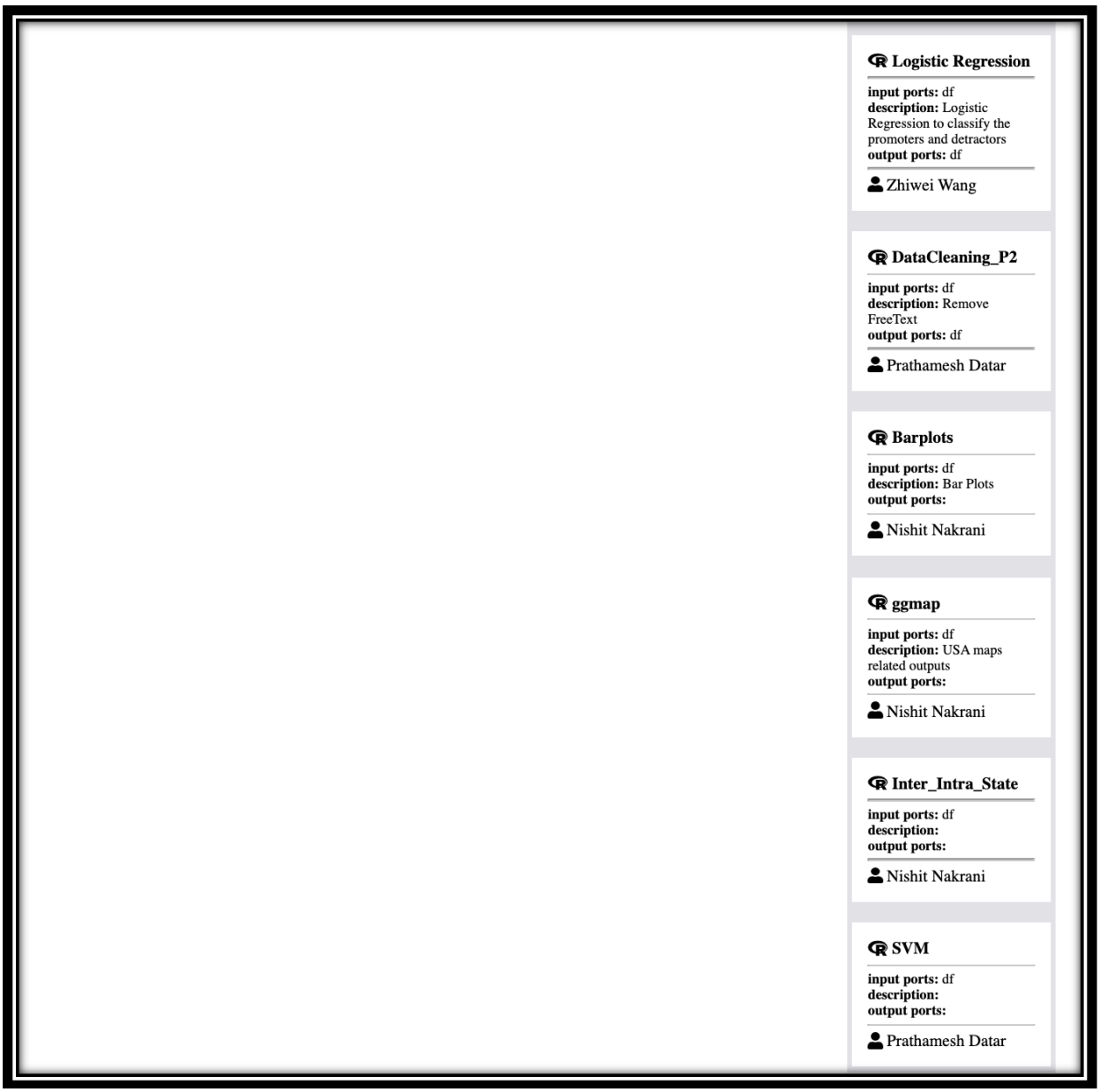
Detractor	Promoter
Customer Service in terms of leg room, luggage handling and overall experience	Customer Service in the way they handle Business Class
Work on facilities and Services for Female, Minors and Senior Citizen	Retain the similar customer service for Male Adults
Improve facilities and services for Customers who fly with Blue Airline Status, Eco Class and Personal Travel	Try to maintain the same quality of service such as for Silver Airline Status, Business Class and Business Travel Customers
Investigate the situation with Flights in Texas especially Dallas and Houston.	Airline can look at Virginia or Chicago and then try to replicate similar customer service throughout other cities and states
Investigate FlyFast Airways and if the situation does not improve discontinue their association with SouthEast Airlines	Increase Association with Airlines like Northwest Business Airlines Inc. which have a good customer reputation

8. MIDST

The tasks were discussed in team meetings and individual tasks were assigned to all. The individuals who were assigned with the tasks moved the tasks across the MIDST, based on the results obtained. Results and analysis were discussed during the team meetings.




Proposed
Not Started
In Progress
Validating
Completed
DataImport
input ports:
description:
output ports: df
Prathamesh Datar
DataCleaning_P1
input ports: df
description: Removing NAs, creating new additional columns
output ports: df
Prathamesh Datar
AssociateRuleMining
input ports: df
description:
output ports:
Pranjal Sondkar
LinearModels
input ports: df
description:
output ports:
Pranjal Sondkar
DataCleanText
input ports: df
description: Dataframe for Text Mining
output ports: df
Prathamesh Datar
TextMining
input ports: df
description: Most frequent words & bigrams in promoter/detractors' freetext(comment) columns
output ports:
Zhiwei Wang


Logistic Regression
input ports: df
description: Logistic Regression to classify the promoters and detractors
output ports: df
Zhiwei Wang
DataCleaning_P2
input ports: df
description: Remove FreeText
output ports: df
Prathamesh Datar
Barplots
input ports: df
description: Bar Plots
output ports:
Nishit Nakrani
ggmap
input ports: df
description: USA maps related outputs
output ports:
Nishit Nakrani
Inter_Intra_State
input ports: df
description:
output ports:
Nishit Nakrani
SVM
input ports: df
description:
output ports:
Prathamesh Datar