IST 687 – INTRODUCTION TO DATA SCIENCE
Lab Section M009 | Group 1

Reducing Customer Churn for Southeast airline

Submitted by:
Jimit Mistry, Sourabh Ghosh, Vasundhara Patil, Sanjana Bhot, Sanyukta Shandilya

TABLE OF CONTENTS

INTRODUCTION

In this project, using different analysis techniques on customer feedback data, a way to lower customer churn for Southeast Airlines is recommended. Southeast airlines believed that the best way to minimize customer churn was to have a loyalty program for frequent customers. However, there was not enough data to support this line of thinking. One of the reasons why just relying on their loyalty program was not sufficient in keeping low customer churn, the customers were valuing the customer loyalty program less. The real goal is to reduce churn by getting ahead of the loss (of the customer) by identifying some leading indicators, or metrics, that might help keep a customer. These insights could provide actionable suggestions as to how to avoid having the customers leave and go to another airline.

NPS asks customers to respond, on a scale of 1 -10, to one basic question of how likely they will recommend the airline to a friend or colleague.

If respondents score less than 7, they're detractors. If they scored above an 8, they're promoters. In the middle range (a score of 7 or 8), then they're "passive". The concept of NPS is that customers who are promoters are good customers. Such customers may sometimes even provide free "word of mouth" advertising. Customers who are detractors are really problematic in that they may actively tell their social connections not to use the product or service. It has often been suggested that NPS provides a good proxy for understanding how likely a customer is to churn.

Southeast Airlines is one of the top four airlines in the United States. Customers buying a Southeast plane ticket fly on Southeast Airlines' primary routes as well as on Southeast's regional partner airlines just like the other top airlines. Regional airlines act as feeder airlines to major airlines by connecting smaller airports to the airline's main hubs. Hub airports are always located in major cities, whereas the regionals serve smaller cities and rural areas. Like other airlines, Southeast contracted out to regional carriers because it allowed them to lower their risks related to capacity and pricing.

The survey dataset contained thousands of observations of flight segment data collected by Southeast Airlines. Each instant represents one flight segment, by one airline, for a specific customer. Each column represents an attribute of that particular flight segment. Each row captures 26 characteristics of the flight, the customer. It should be noted that there are some missing values in the dataset.

BUSINESS QUESTIONS

• Which characteristics contribute towards making the customer a Promoter or a Detractor for the airline?

• In which areas does the airline need to improve, and how can it improve to reduce customer churn?

• How can the customer be changed from a Detractor or a Passive customer to a Promoter?

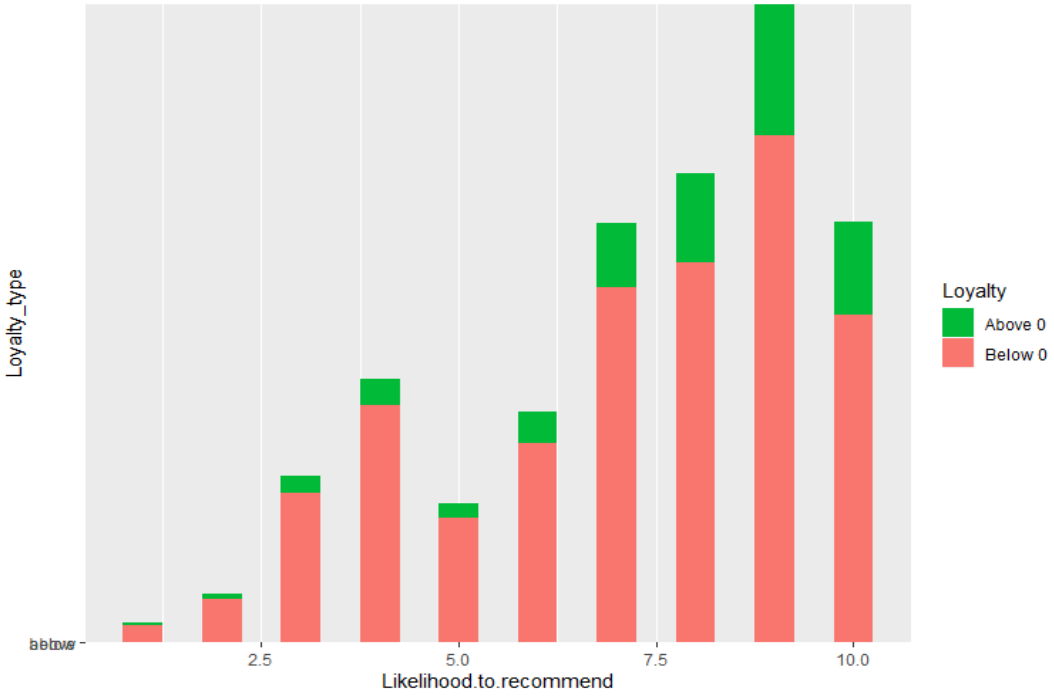
• What facilities and services the airline needs to focus on to increase the overall satisfaction rate?

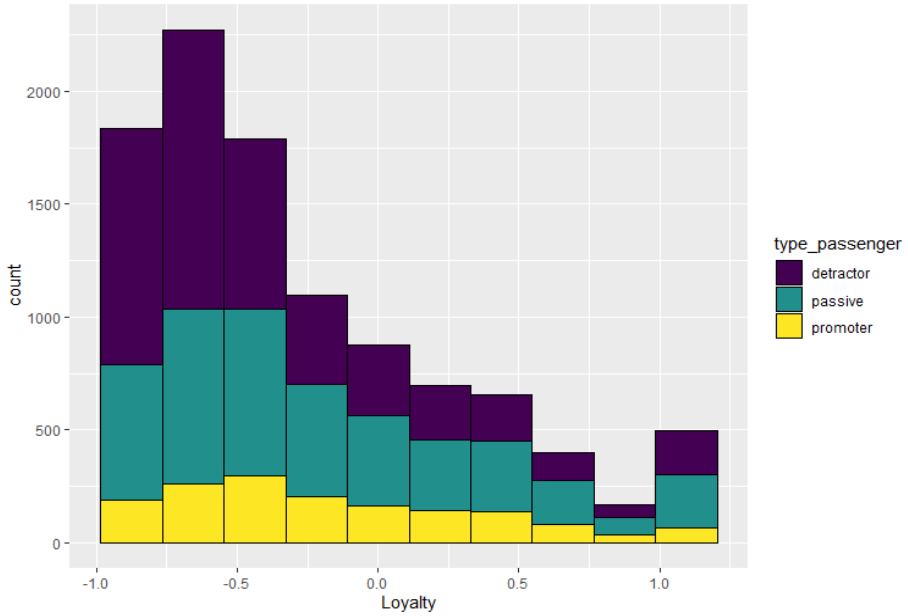
These Business questions were answered by following the below action plan:

- Identifying the target labels(Detractors and Promoters) and visualizing it as a classification problem.
- Data cleaning
- Exploratory Data Analysis was implemented to understand and generate distribution graphs
- Feature Engineering and Data transformations
- Implemented classification and regression models to get actionable insights.
- Conclusion and Inferences were drawn based on characteristics features.

DESCRIPTIVE ANALYSIS AND DATA VISUALIZATION

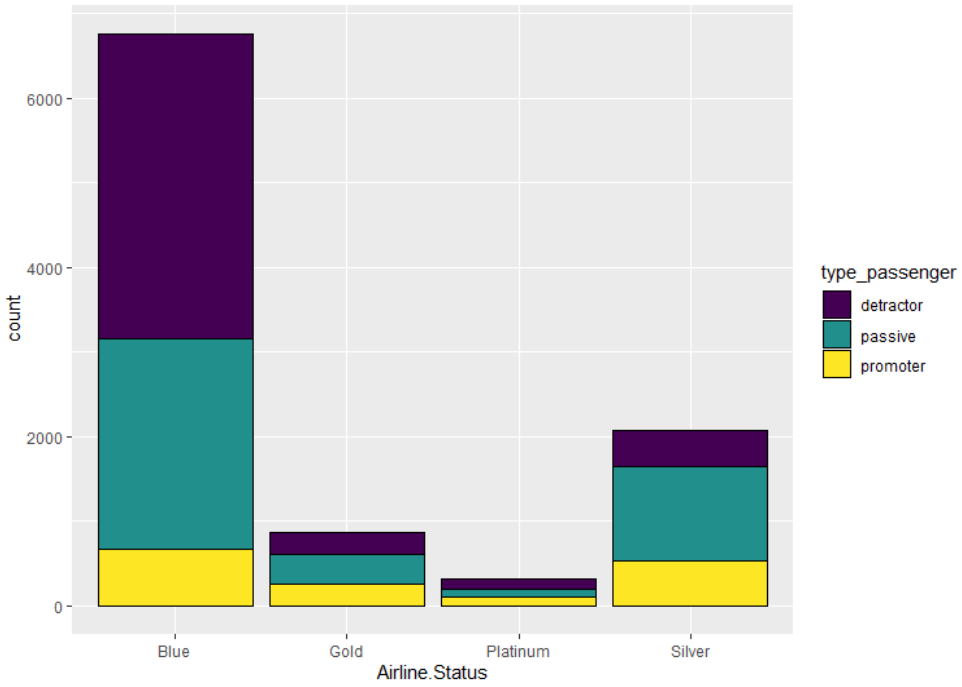
Loyalty Score

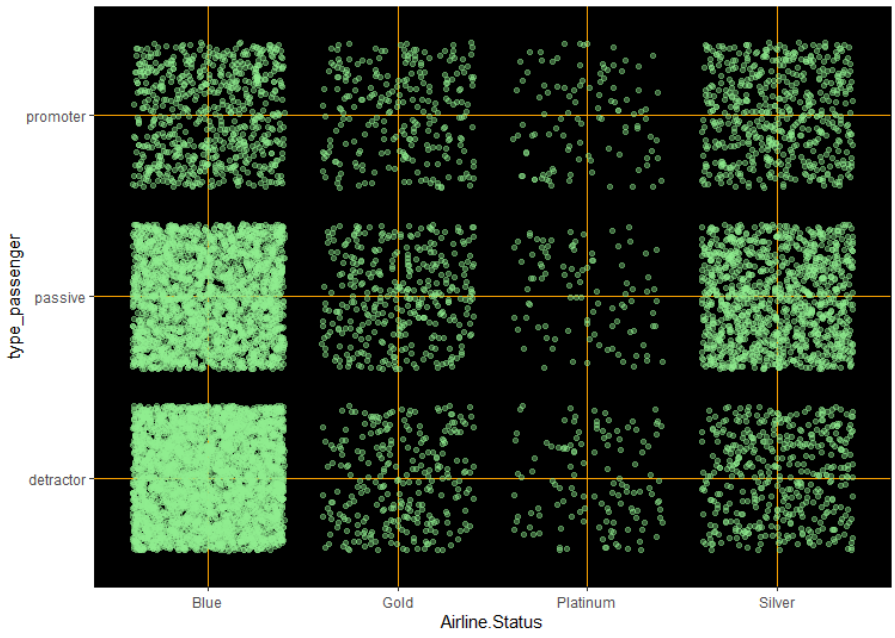




An index of loyalty ranging from -1 to 1 that reflects the proportion of flights taken on other airlines versus flight taken on this airline. A higher index means more loyalty.

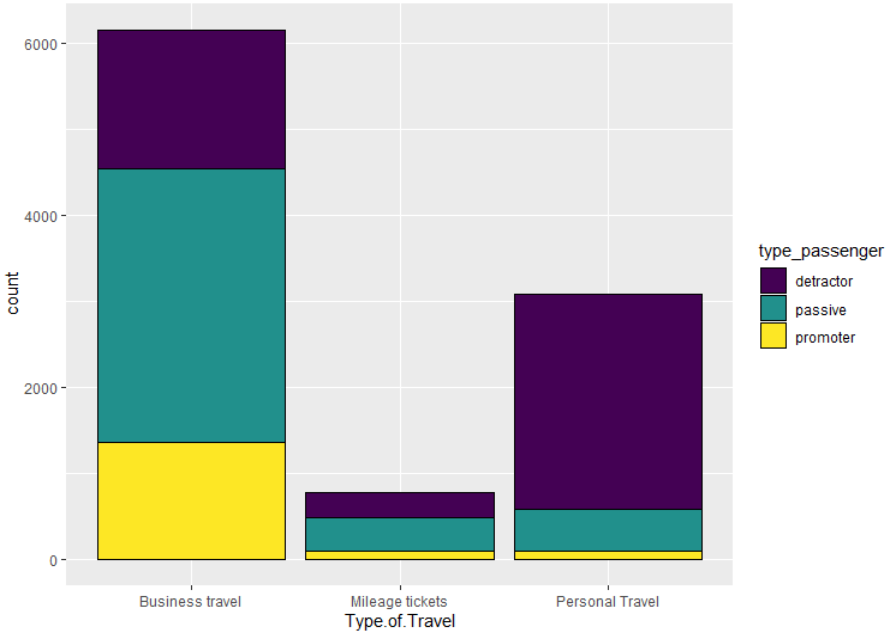
Airline Status



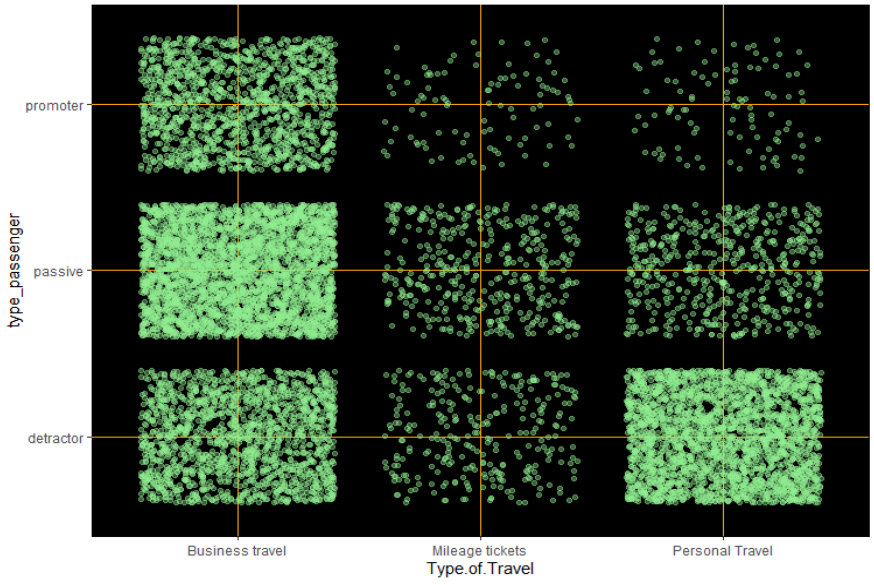


From the above bar plot, we can see that the count of detractors is more in Blue Airline status. We can see the same result in the density plot graph.

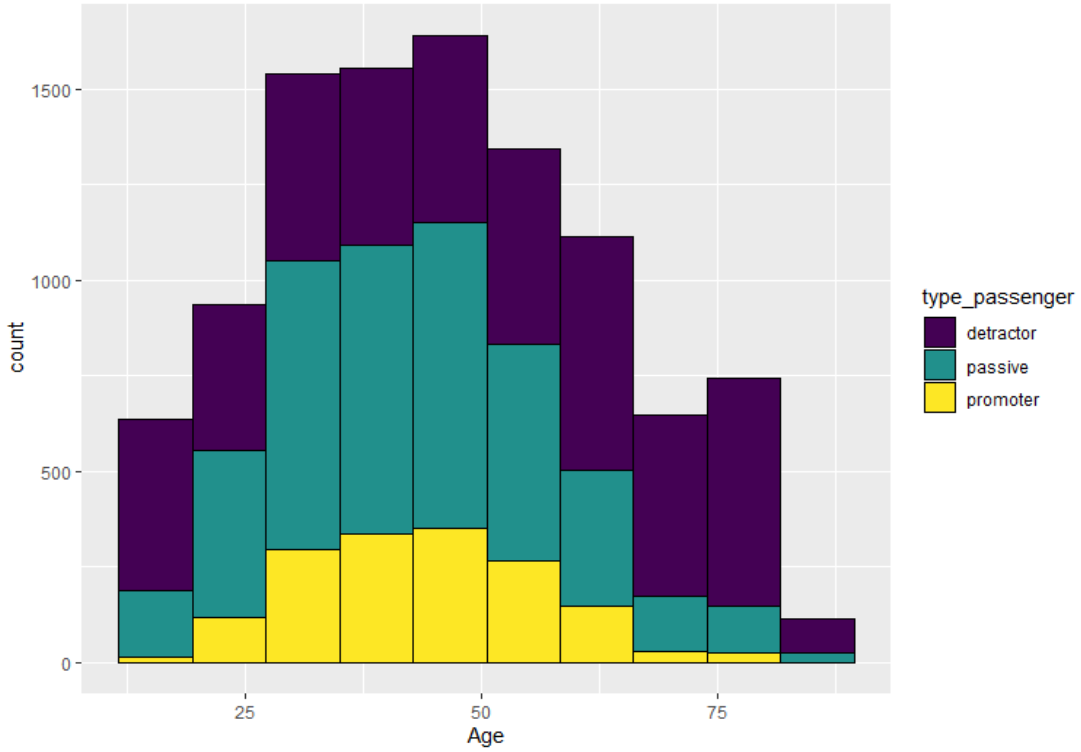
Type of Travel

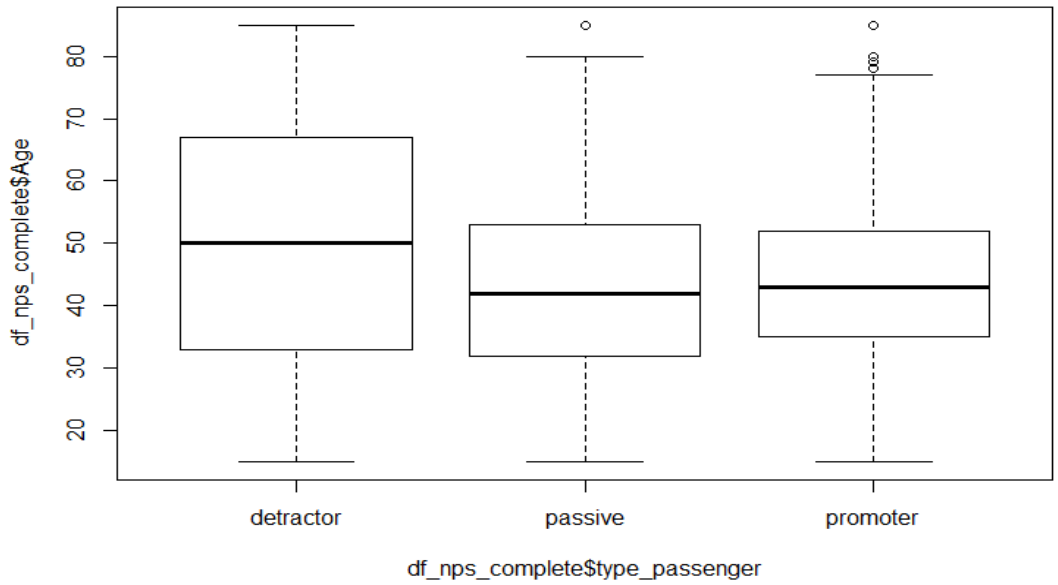


The above graph shows the type of travel vs the count. We can see that the count of detractors is more with people using personal travel. The promoter count for personal travel is very less which needs to be improved.



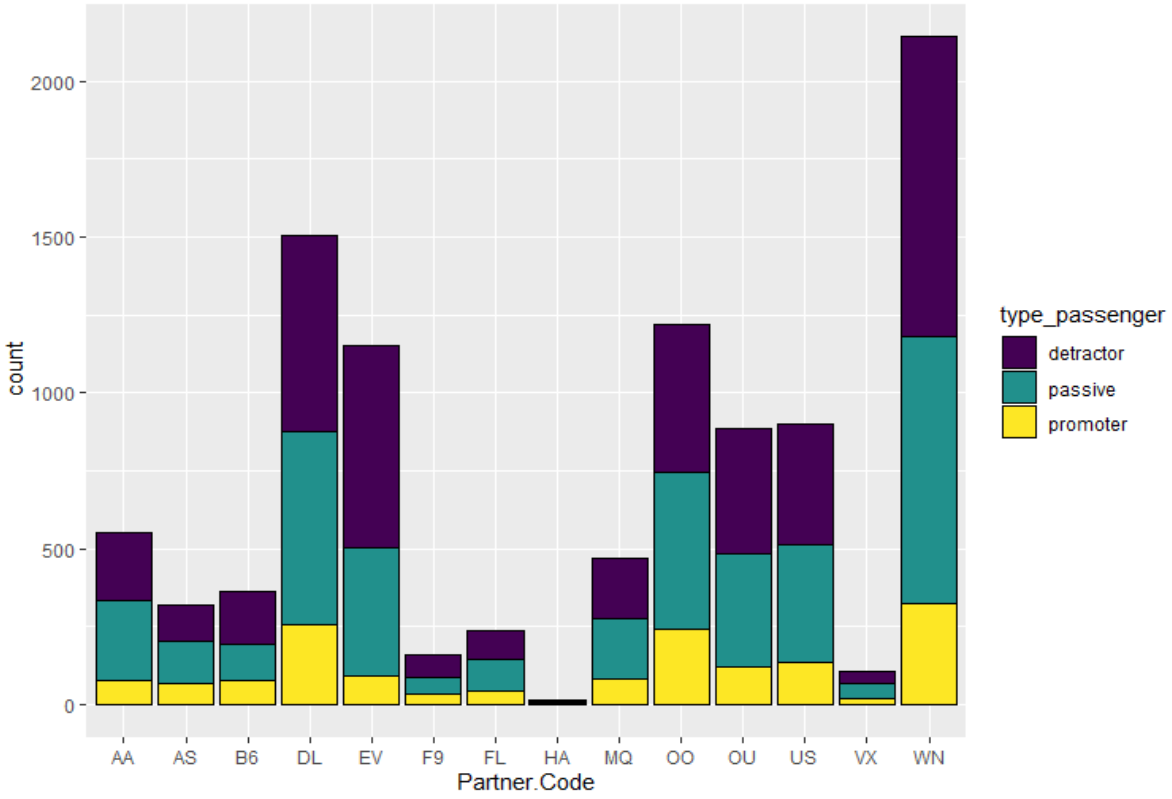
Age


1500
1000
500
0
count
25
50
75
Age
type_passenger
detractor
passive
promoter


df_nps_complete$Age
20
30
40
50
60
70
80
detractor
passive
promoter
df_nps_complete$type_passenger

From the above graph, we can see that customers of age 50 and above have less count of promoters. This can also be seen in the box plot graph where the median for detractor is 50.

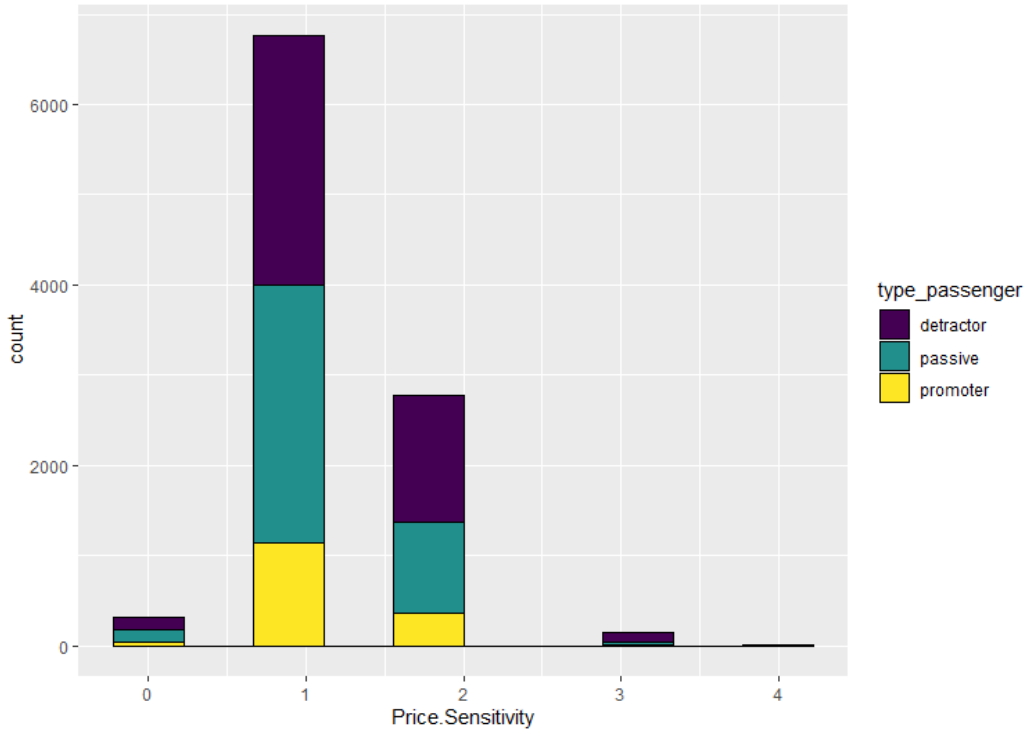
Partner Name



From the above graph, we can see that Partner Code HA has only detractors. Also, the Partner Code WN has the most number of detractors compared to others.

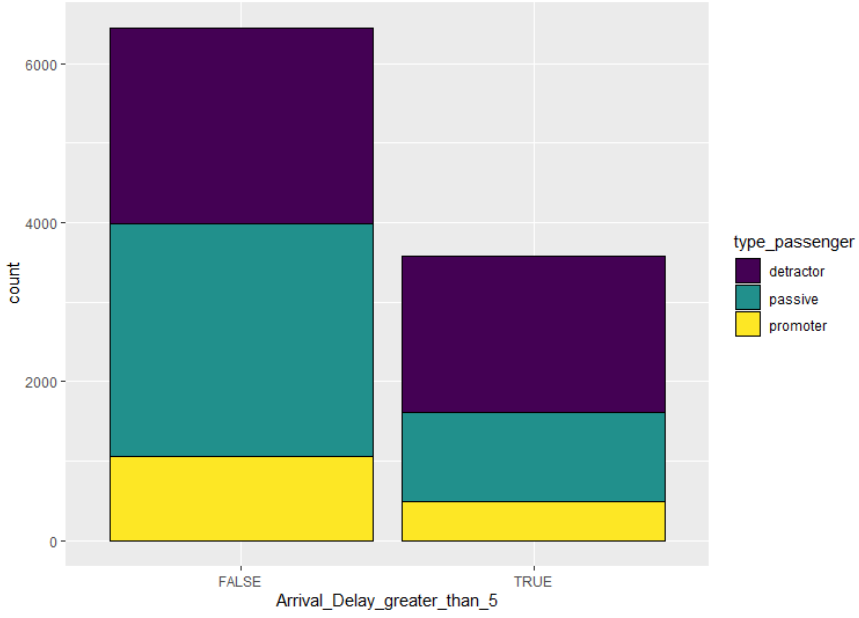
Price Sensitivity

From the above graph, we can see that the maximum number of customers are detractors and passive customers at level 1. And when we compare level 1 and level 2, we see that the number



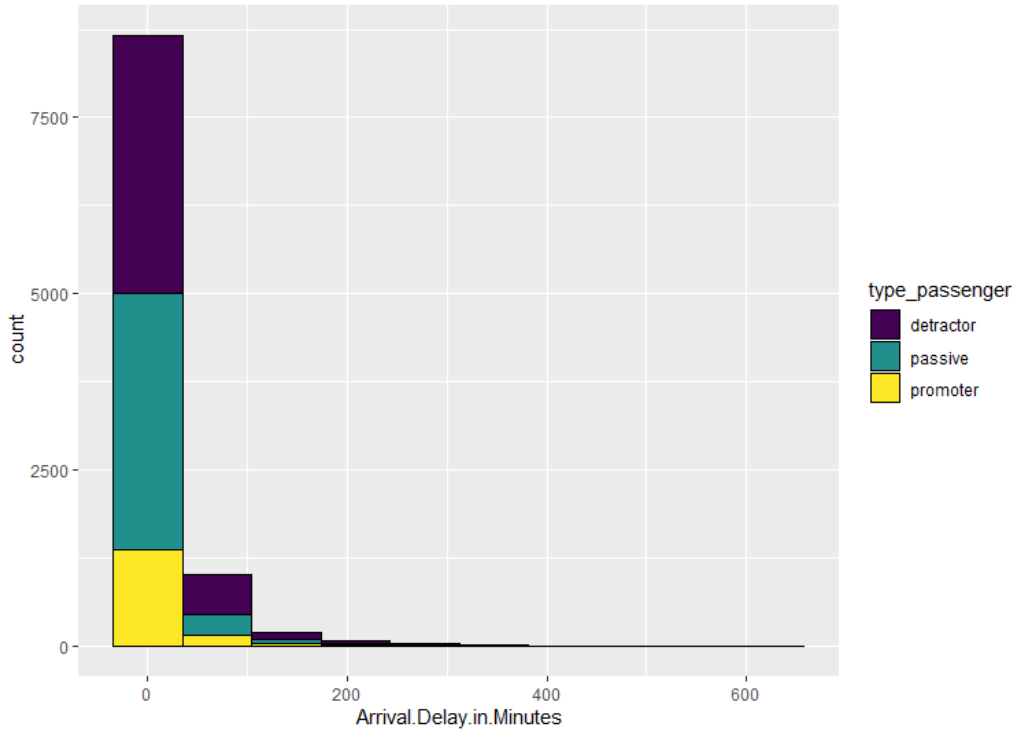
of promoters has reduced drastically. In level 3 we can only see detractors and passive customers.

Arrival Delay



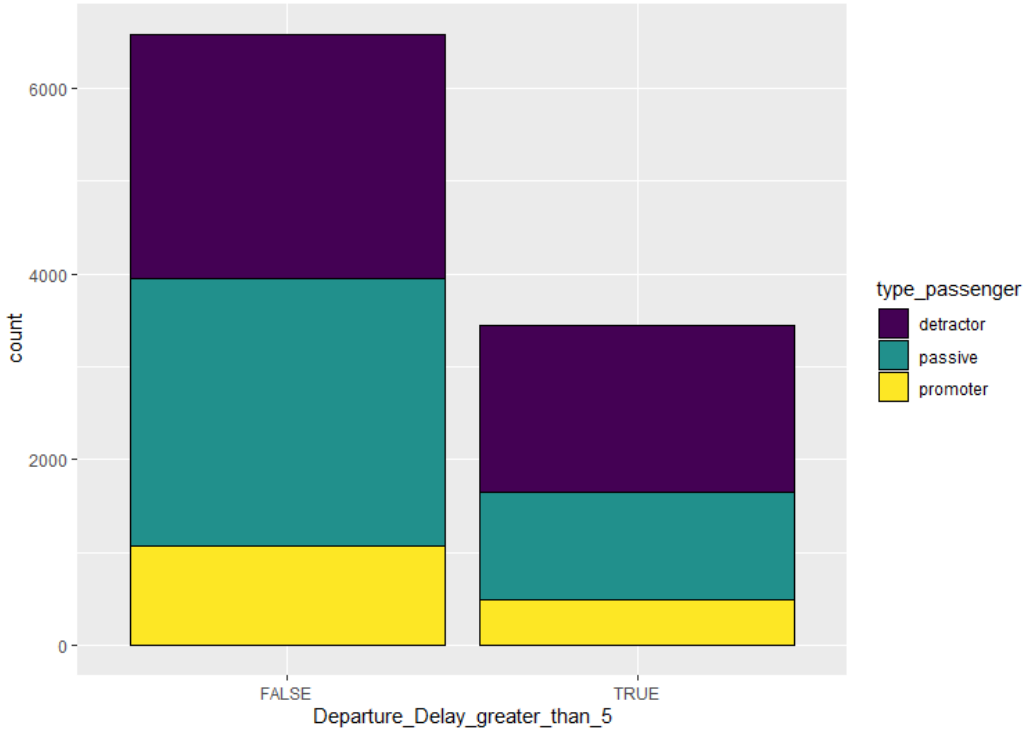
z

From the Arrival delay graph, we can infer that when the arrival delay is less than 5, we have more promoters and fewer detractors and passive passengers. However, if the flight is delayed by more than 5 minutes there are more detractors and very few promoters.

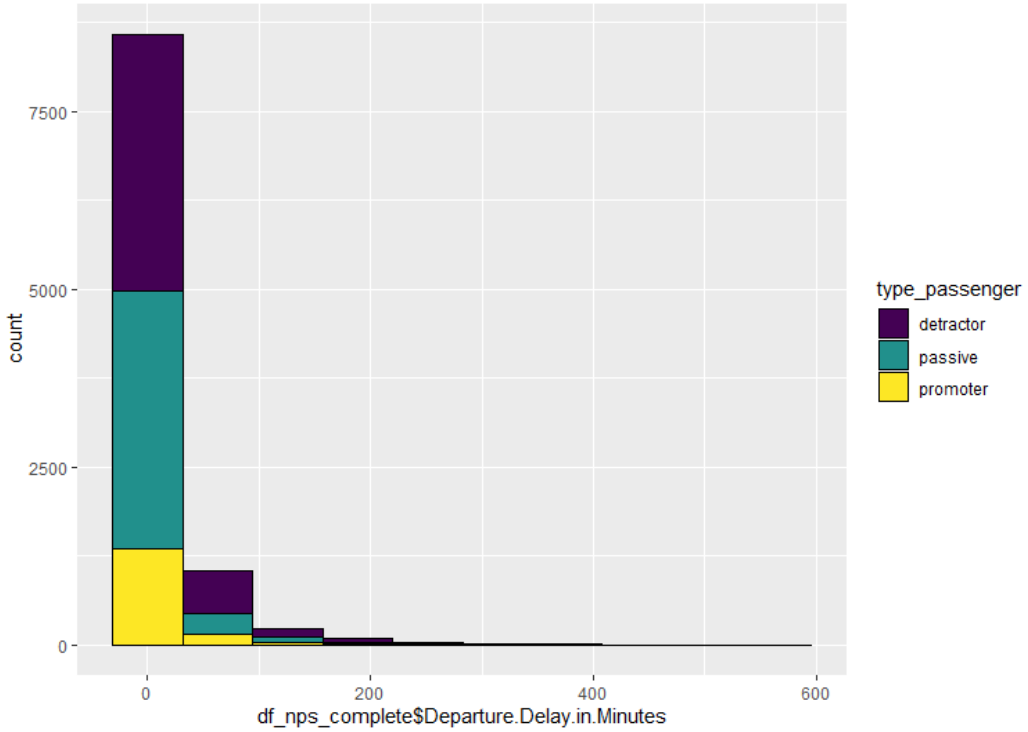


Here we see that, as the arrival delay minutes increase the number of detractors increase and the number of promoters is decreasing significantly.

Departure Delay

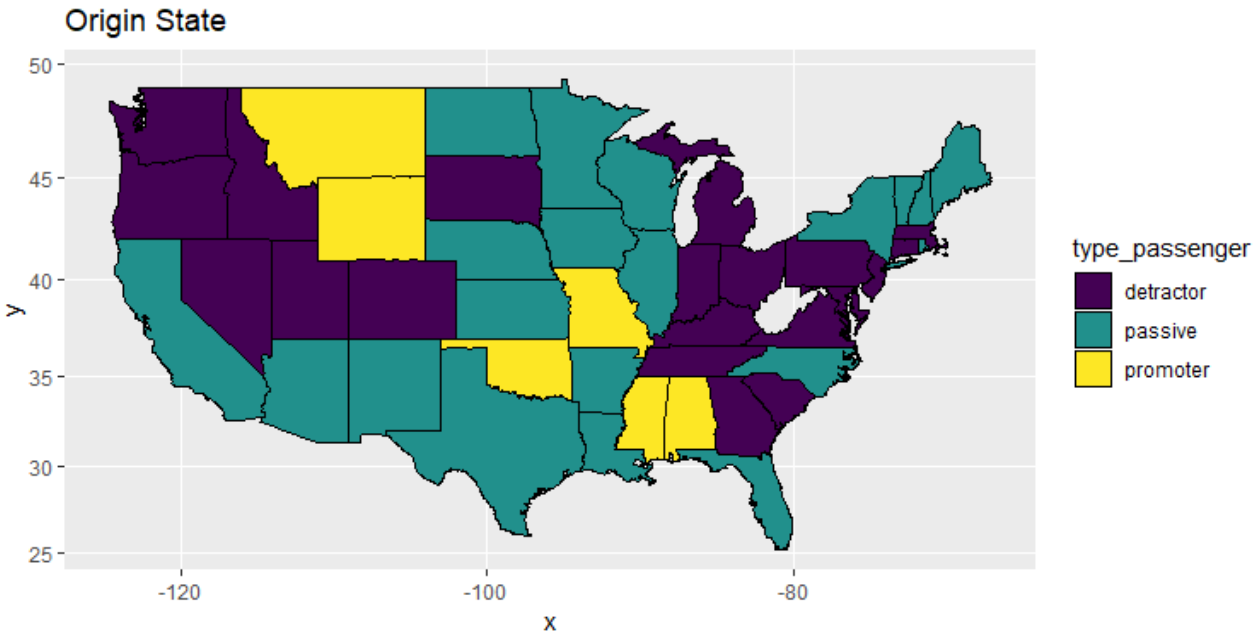


From the Departure delay graph, we can infer that when the departure delay is less than 5 then we have more promoters compared to when the flight departs more than 5 minutes late.

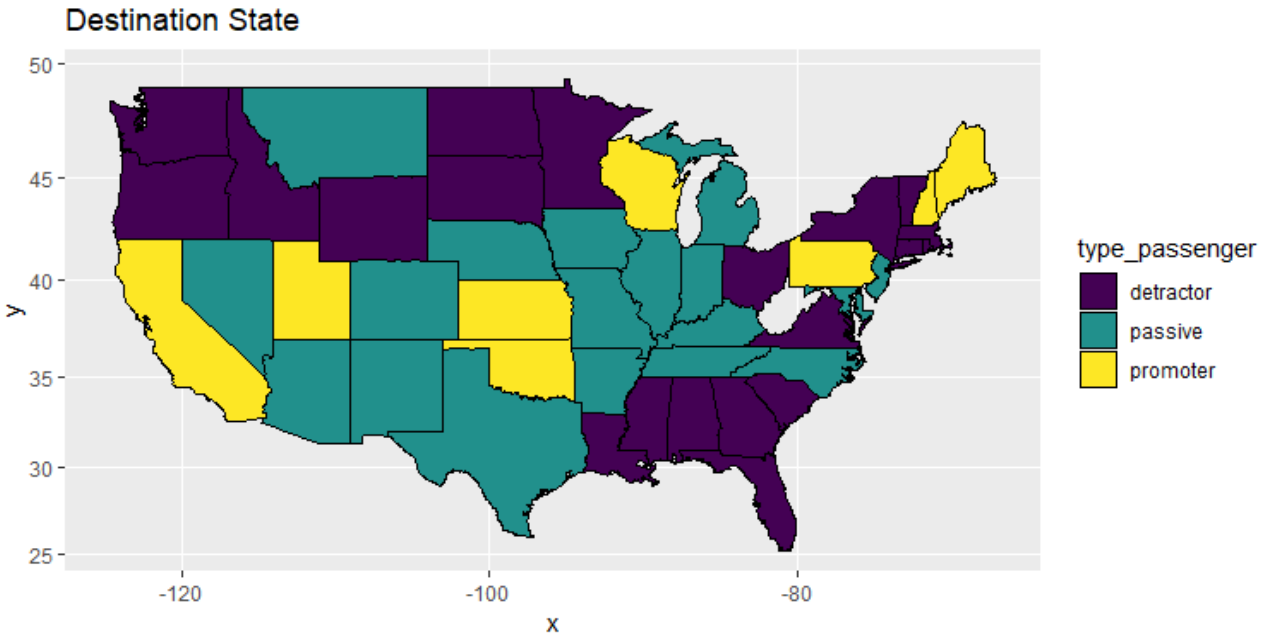


Here we see that, as the arrival delay minutes increase the number of detractors increase.

State-wise analysis



The states in purple are the maximum detractors of flights departing from those states and in yellow are promoters.



The states in purple are the maximum detractors of flights arriving from those states and in yellow are promoters.

Models Used

After analysis of all the attributes of the dataset, a full proof method is needed to identify the most important attributes out of all, to identify the bottlenecks. This process can be done by building models that also have verification metrics like accuracy. Our dependent variable or the feature that we want to calculate is Likelihood.to.recommend or type_passenger which contains the information of “Detractor”, “Passive” and “Promoter”.

Data Transformation

The raw data provided was not suitable to be used in the models, because of the algorithmic requirements of specific data types, and because of differences in their scales. The solution was to convert every numerical attribute to categorical values by splitting them into inappropriate parts, based on information obtained about the distribution through histograms. That means that we put smaller cuts in the region where the majority of data lied, and put larger cuts where data points were sparse, thus avoiding any important information loss, to gain better insights. The method used for categorizing data is known as **Bucketing**.

```
quantile(df$Age,c(0,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1))
df$Age<-  ifelse(df$Age<=24,'<24',
                                ifelse(df$Age<=35,'25-35',
                                       ifelse(df$Age<=46,'36-46',
                                              ifelse(df$Age<=58,'47-58',
                                                     ifelse(df$Age<=71,'57-71','71+')))))
table(df$Age)

quantile(df$Flights.Per.Year,c(0,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1))
df$Flights.Per.Year<-  ifelse(df$Flights.Per.Year<= 6 ,'<6',
                    ifelse(df$Flights.Per.Year<=12,'6-12',
                           ifelse(df$Flights.Per.Year<=25,'13-25',
                                  ifelse(df$Flights.Per.Year<=38  ,'26-38','38+'))))
table(df$Flights.Per.Year)

quantile(df$Loyalty,c(0,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1))
df$Loyalty<-  ifelse(df$Loyalty<= -0.75 ,'<-0.75',
                                ifelse(df$Loyalty<=-0.45,'-0.7501 to -0.45',
                                       ifelse(df$Loyalty<=0.0588,'-0.451 to 0.0588',
                                              ifelse(df$Loyalty<=0.3767  ,'0.059 to 0.3768','0.3768+'))))
table(df$Loyalty)

quantile(df$Total.Freq.Flyer.Accts,c(0,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1))
df$Total.Freq.Flyer.Accts<-  ifelse(df$Total.Freq.Flyer.Accts==0 ,'0',
                     ifelse(df$Total.Freq.Flyer.Accts==1,'1','1+'))

table(df$Total.Freq.Flyer.Accts)

quantile(df$Departure.Delay.in.Minutes,c(0,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1),na.rm=TRUE)
df$Departure.Delay.in.Minutes<-  ifelse(df$Departure.Delay.in.Minutes<=5 ,'LESS THAN 5 MINS',
                                        ifelse(df$Departure.Delay.in.Minutes<=10 ,'5 MINS TO 10 MINS',
                                               ifelse(df$Departure.Delay.in.Minutes<=20 ,'10 MINS TO 20 MINS',
                                                      ifelse(df$Departure.Delay.in.Minutes<=60 ,'20 MINS TO 60 MINS',
                                                             ifelse(df$Departure.Delay.in.Minutess<=180,'60 MINS TO 180 MINS','180+ Minutes')))))

table(df$Departure.Delay.in.Minutes)
df$Departure.Delay.in.Minutes[which(is.na(df$Departure.Delay.in.Minutes)==TRUE)]<-'0-60 Mins'

quantile(df$Arrival.Delay.in.Minutes,c(0,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1),na.rm=TRUE)
df$Arrival.Delay.in.Minutes<-  ifelse(df$Arrival.Delay.in.Minutes<=5 ,'LESS THAN 5 MINS',
                                      ifelse(df$Arrival.Delay.in.Minutes<=10 ,'5 MINS TO 10 MINS',
                                             ifelse(df$Arrival.Delay.in.Minutes<=20 ,'10 MINS TO 20 MINS',
                                             ifelse(df$Arrival.Delay.in.Minutes<=60 ,'20 MINS TO 60 MINS',
                                             ifelse(df$Arrival.Delay.in.Minutes<=180,'60 MINS TO 180 MINS','180+ Minutes')))))

table(df$Arrival.Delay.in.Minutes)
df$Arrival.Delay.in.Minutes[which(is.na(df$Arrival.Delay.in.Minutes)==TRUE)]<-'0-60 Mins'
```

```
quantile(df$Flight.time.in.minutes,c(0,0.06,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1),na.rm=TRUE)
df$Flight.time.in.minutes<-  ifelse(df$Flight.time.in.minutes<= 40,'LESS THAN 40 MINS',
                                    ifelse(df$Flight.time.in.minutes<=180 ,'40 MINS TO 180 MINS',
                                           ifelse(df$Flight.time.in.minutes<=230,'180 MINS TO 230 MINS', '230+ MINS')))


table(df$Flight.time.in.minutes)
df$Flight.time.in.minutes[which(is.na(df$Flight.time.in.minutes)==TRUE)]<-'230 MINS'

quantile(df$Flight.Distance,c(0,0.067,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1),na.rm=TRUE)
quantile(df$Flight.Distance, c(0.067, 0.2, 0.33, 0.40, 0.60, 0.80,1))
df$Flight.Distance<-  ifelse(df$Flight.Distance<=200  ,'LESS THAN 200 MILES',
                             ifelse(df$Flight.Distance<=1000,'200 MILES TO 1000 MILES', '1000+ MILES'))



table(df$Flight.Distance)
df$Flight.Distance[which(is.na(df$Flight.Distance)==TRUE)]<-'400 MILES TO 1000.0 MILES'

df_new<-df[,c("Partner.Name","Age","Gender","Airline.Status","Price.Sensitivity",
              "Loyalty","Type.of.Travel","Total.Freq.Flyer.Accts","Class","Flights.Per.Year","Departure.Delay.in.Minutes","Arrival.Delay.in.Minutes"
              ,"Flight.time.in.minutes","Flight.Distance","Flight.cancelled","Likelihood.to.recommend")]

df_new$Departure.Delay.in.Minutes[which(df_new$Flight.cancelled=='Yes')] <- '0'
df_new$Arrival.Delay.in.Minutes[which(df_new$Flight.cancelled=='Yes')] <- '0'
#View(df_new)
na_values <- apply(apply(df_new, 2, is.na), 2, which)
na_values
write.csv(df_new,"data_new2.csv")
```

The variables that had many multiple levels were removed, like Date, Latitude/Longitudes, and Cities/States to avoid model ambiguity. This way the model would be able to converge on some specific values for the attributes.

Every attribute was converted to factors. Finally, a total of 16 variables were left at the end of this step.

```
> glimpse(df_new)
Observations: 10,282
Variables: 16
$ Partner.Name               <fct> Northwest Business Airlines Inc., Cheapseats Airlines Inc., EnjoyFlying Air Services, Oursin Ai...
$ Age                        <fct> 36-46, 25-35, 47-58, 57-71, 36-46, 36-46, 47-58, 47-58, 57-71, 25-35, <24, 57-71, 47-58, 25-35,...
$ Gender                     <fct> Female, Male, Female, Female, Female, Male, Female, Female, Female, Female, Female, Male, Male,...
$ Airline.Status             <fct> Gold, Gold, Blue, Blue, Blue, Gold, Gold, Blue, Blue, Silver, Blue, Blue, Silver, Gold, Silver,...
$ Price.Sensitivity          <int> 1, 2, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1,...
$ Loyalty                    <fct> 0.3768+, -0.451 to 0.0588, 0.3768+, -0.7501 to -0.45, 0.3768+, 0.3768+, -0.451 to 0.0588, -0.75...
$ Type.of.Travel             <fct> Business travel, Business travel, Personal Travel, Business travel, Personal Travel, Business t...
$ Total.Freq.Flyer.Accts     <fct> 1+, 0, 0, 1+, 1+, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1+, 0, 0, 1+, 1+, 1+, 1+, 1, 1, 0, 0, 1+, 1, 0, 0,...
$ Class                      <fct> Business, Business, Eco, Business, Eco, Eco, Eco, Eco Plus, Eco Plus, Eco, Eco, Eco, Eco, Eco, ...
$ Flights.Per.Year           <fct> <6, 13-25, <6, 38+, 6-12, <6, 6-12, 38+, 26-38, 38+, 6-12, 38+, <6, 13-25, 26-38, 13-25, 38+, 1...
$ Departure.Delay.in.Minutes <fct> LESS THAN 5 MINS, LESS THAN 5 MINS, LESS THAN 5 MINS, LESS THAN 5 MINS, LESS THAN 5 MINS, 5 MIN...
$ Arrival.Delay.in.Minutes   <fct> LESS THAN 5 MINS, LESS THAN 5 MINS, LESS THAN 5 MINS, LESS THAN 5 MINS, LESS THAN 5 MINS, LESS ...
$ Flight.time.in.minutes     <fct> 40 MINS TO 180 MINS, 40 MINS TO 180 MINS, 40 MINS TO 180 MINS, 180 MINS TO 230 MINS, LESS THAN ...
$ Flight.Distance            <fct> 200 MILES TO 1000 MILES, 200 MILES TO 1000 MILES, 200 MILES TO 1000 MILES, 1000+ MILES, LESS TH...
$ Flight.cancelled           <fct> No, No, No, No, No, No, No, No, No, No, No, Yes, No, No, No, No, No, No, No, No, No, No, No, No...
$ Likelihood.to.recommend    <int> 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0,...
```

Feature Importance Generation to find the most significant Features

In order to get optimum Feature Importance, an incremental modeling technique was implemented. The following models and strategies were implemented:

- Logistic Regression
- Random Forest Classifier

· XGBoost Classifier

Intermediate Step: The Relative Importance of attributes or features in each of the models were compared to figure out the best combination of variables. These features were used in a Rule-based algorithm to get the exact values for those features for the desired output of a "Detractor" and a "Promoter".

· Association Rule Mining (Apriori) Algorithm

Logistic Regression

Regression analysis is a set of statistical processes for estimating the relationships between a dependent variable and one or more independent variables [1]. As our output of Likelihood.to.recommend is basically discrete values, Logistic regression is the most favorable model. Logistic regression works on binary output, "yes" and "no", so we categorized out Likelihood.to.recommend into 1 for promoters, and 0 for passive and detractors.

```
df$Likelihood.to.recommend[(df$Likelihood.to.recommend<7)] <- 0 # 'Detractor'
df$Likelihood.to.recommend[(df$Likelihood.to.recommend==7)] <- 0 #'Passive'
df$Likelihood.to.recommend[(df$Likelihood.to.recommend==8)] <- 0# 'Passive'
df$Likelihood.to.recommend[(df$Likelihood.to.recommend==9)] <- 1# 'Promoter'
df$Likelihood.to.recommend[(df$Likelihood.to.recommend==10)] <-1# 'Promoter'
df$Likelihood.to.recommend<-as.factor(df$Likelihood.to.recommend)
```

The next step is to create a train and test datasets in order to validate our model by calculating accuracy and AUC. The glm() function of the caTools library was used to generate a Logistic regression model.

```
library(caTools)

# Create train and test data sets
set.seed(88)
split <- sample.split(df_new$Likelihood.to.recommend, SplitRatio = 0.75)
train <- subset(df_new, split == TRUE)
test <- subset(df_new, split == FALSE)

#Logistic model
model <- glm(Likelihood.to.recommend ~.,family=binomial(link='logit'),data=train)
summary(model)
anova(model, test="Chisq")

fitted.results <- predict(model,newdata=subset(test),type='response')
fitted.results <- ifelse(fitted.results > 0.5,1,0)

misClasificError <- mean(fitted.results != test$Likelihood.to.recommend,na.rm=TRUE)
print(paste('Accuracy',1-misClasificError))

###"Accuracy 0.756420233463035"
y_train <- dresstrain$Likelihood.to.recommend

x_train <-dresstrain%>% select(Partner.Name,Age,Gender,Airline.Status,Price.Sensitivity,Loyalty,Type.of.Travel,Total.Freq.Flyer.Accts
                               ,Class,Flights.Per.Year,Departure.Delay.in.Minutes,Arrival.Delay.in.Minutes,Flight.time.in.minutes
                               ,Flight.Distance,Flight.cancelled) %>% data.matrix()
library(ROCR)
p <- predict(model, newx=subset(x_train), type="response")
pr <- prediction(p, y_train)
auc <- performance(pr, measure = "auc")
auc
#0.8409551
```

Finally, the last step included calculating accuracy and AUC against the test dataset. The accuracy obtained is 75.642%, which determines that the model predicts well for a promoter or a detractor. The AUC was 84.095% which means that our model works well on any other dataset as well. Thus, our model is a good predictor, as well as generalized, thus can work on any other dataset.

Thus, it clarifies that the features that the model uses are good predictors, and the important features are obtained from the Beta values or the coefficients of the predictors and their p-values.

But, the model can be improved with hyperparameter tuning. Therefore, tune grid was used for hyperparameter tuning and used cross-validation to obtain the best hyperparameters for the model. In logistic regression, that hyperparameter is known as lambda. The best lambda for the model is the minimum value of lambda, which we obtained after cross-validation.

Random Forest Regression

The random forest algorithm works by aggregating the predictions made by multiple decision trees of varying depth. Random forest works using the process of bagging and bootstrapping decreasing variance. The algorithm performs well for the classification data. The requirement of using Random forest in the airline data was to extract the feature importance of the given data so as to narrow down our analysis to the most important features affecting customer churning.

The model was trained and measures using train and test data set with a split ratio (75%, 25%).

Hyperparameter tuning was performed using tune grid on ‘n’ trees to extract the best model which performed with 200 trees.

```
#####Random forest model after parameter tuning the best model performs with 200 trees
rf1 <- randomForest(
  Likelihood.to.recommend ~ .,
  ntree = 200,
  data = train,
  nodesize = 1,
  replace = FALSE,
  importance = TRUE
)
#printing output of random forest
print(rf1)
```

The library randomForest was used for implementing the Random forest model.

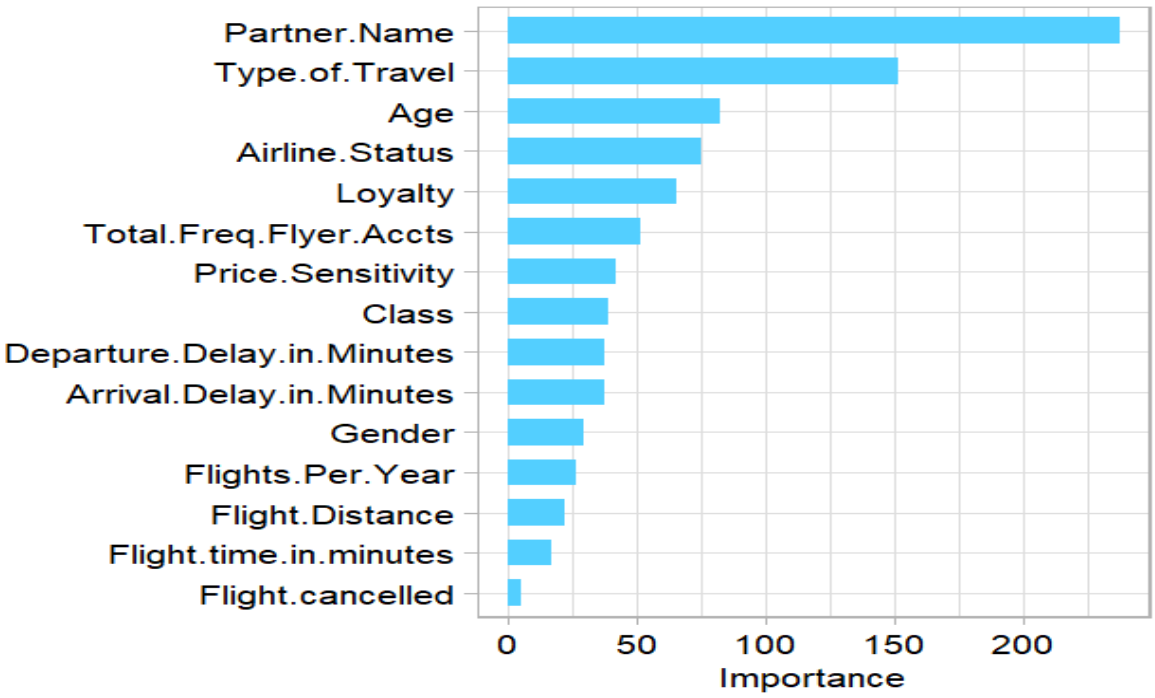
Accuracy and AUC against the test dataset to compare the generalization performance of models. The AUC and Accuracy of the best model were computed to be 84.71% and 76.10% respectively.

```
#Calculating AUC for the model
library(ROCR)
p <- predict(rf1, newdata=subset(test), type="response")
pr <- prediction(p, test$Likelihood.to.recommend)
auc <- performance(pr, measure = "auc")
print(auc)
#AUC 0.8471364

#calculating Accuracy of the model
fitted.results_rf <- predict(rf1,newdata=subset(test),type='response')
fitted.results_rf <- ifelse(fitted.results_rf > 0.4,1,0)

misClasificError <- mean(fitted.results_rf != test$Likelihood.to.recommend,na.rm=TRUE)
print(paste('Accuracy',1-misClasificError))
#"Accuracy 0.761089494163424"
```

Feature Importance after Random Forest (output) :



XGBoost Classifier

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. It is an ensemble learner that builds a strong prediction model as an ensemble of a weak prediction model. The usage of XGBoost in our course of action as it removes correlated

features, this algorithm is a special case of boosting since it functions with gradient descent for better performance.

This model was implemented to compare the relative importance of features evaluated by the Random Forest algorithm.

Feature engineering of data was done for this algorithm converting input features into a sparse matrix.

The library xgboost was used for implementing the XGBoost model.

```
#feature engineering for xgboost converting features into sparse matrix
sparse_matrix <- sparse.model.matrix(Likelihood.to.recommend~.-1, data = train)
output_vector = train$Likelihood.to.recommend

#xgboost model
bst <- xgboost(data = sparse_matrix, label = output_vector, max.depth = 4,
               eta = 1, nthread = 2, nrounds = 10,objective = "binary:logistic")
```

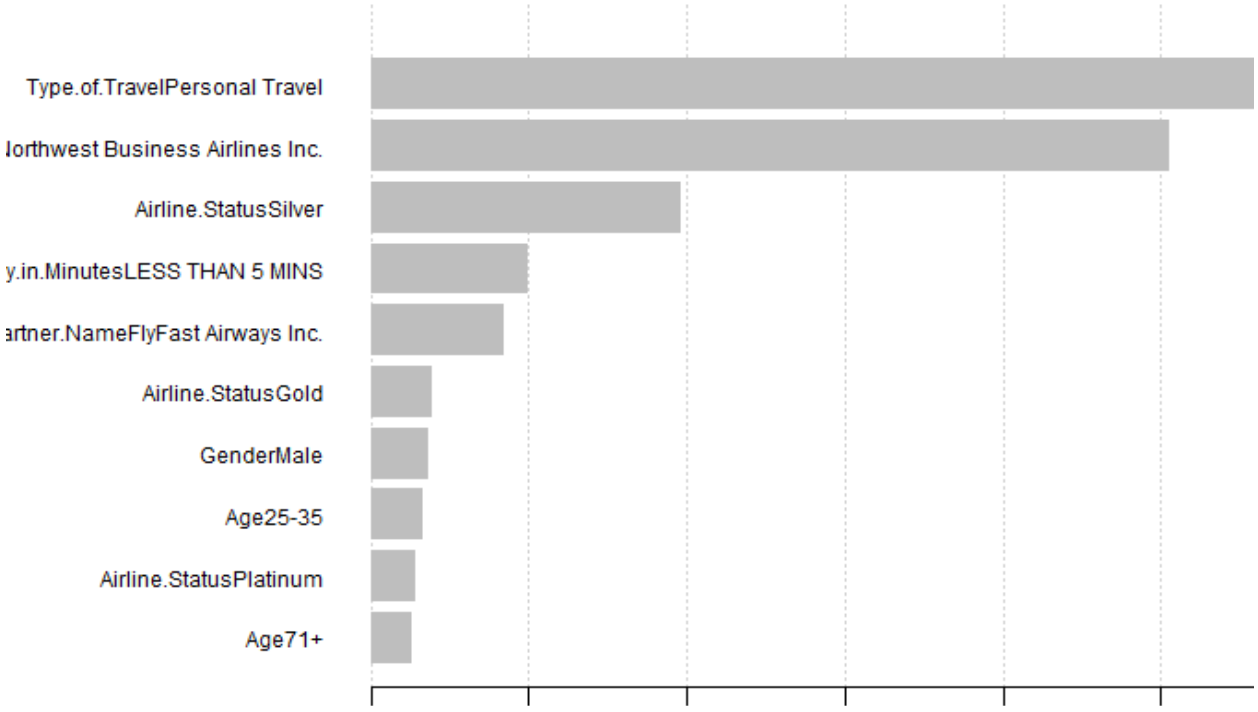
Accuracy and AUC against the test dataset to compare the generalization performance of models. The AUC and Accuracy of the best model were computed to be 83.78% and 75.44% respectively.

```
#Calculating generalization performance measue AUC
sparse_matrix_test <- sparse.model.matrix(Likelihood.to.recommend~.-1, data = test)
output_vector_test = test$Likelihood.to.recommend
pred <- predict(bst, sparse_matrix_test)
pr <- prediction(pred, output_vector_test)
auc <- performance(pr, measure = "auc")
print(auc)
#0.8378625

#Calculating model Accuracy
fitted.results_gbm <- predict(bst,sparse_matrix_test)
fitted.results_gbm <- ifelse(fitted.results_gbm > 0.5,1,0)

misClasificError <- mean(fitted.results_gbm != test$Likelihood.to.recommend,na.rm=TRUE)
print(paste('Accuracy',1-misClasificError))
#"Accuracy 0.754474708171206
```

Feature Importance after XGBoost (output) :

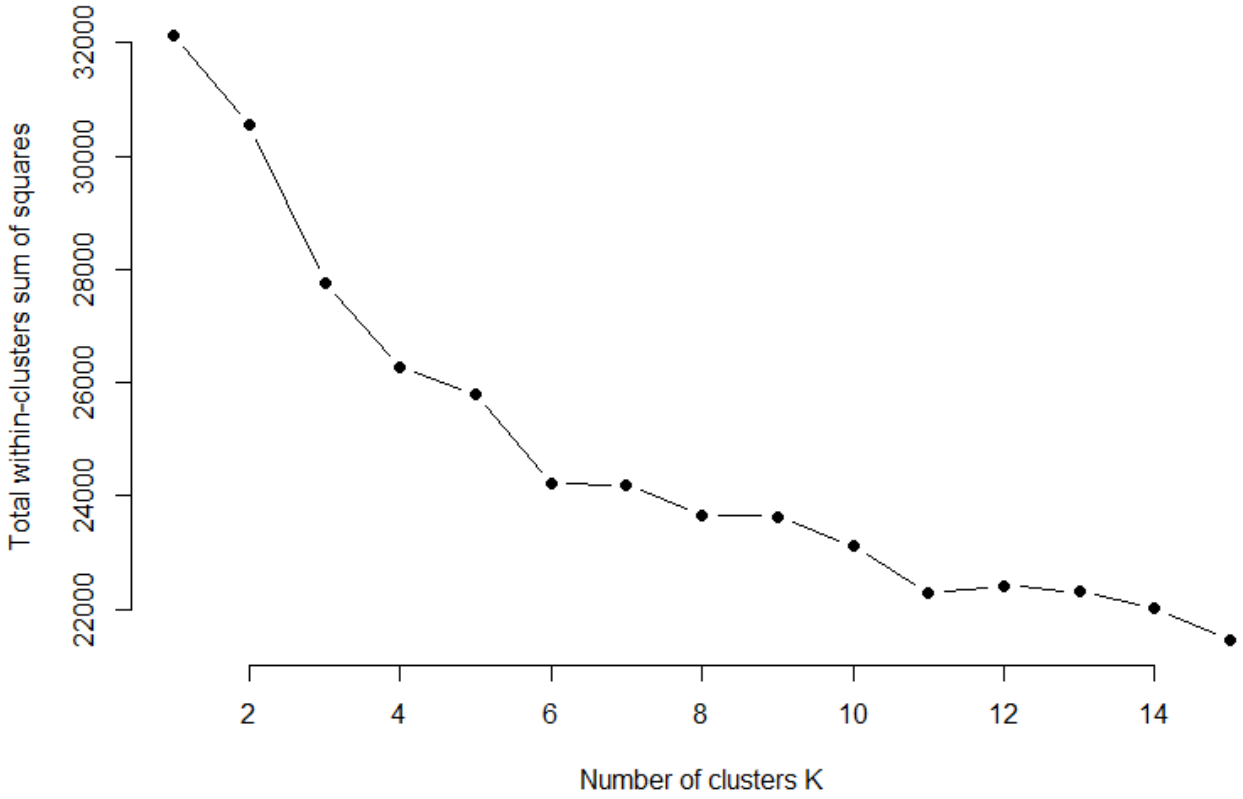


RESULTS AND INFERENCES BASED ON MODELS

K-Prototype Clustering

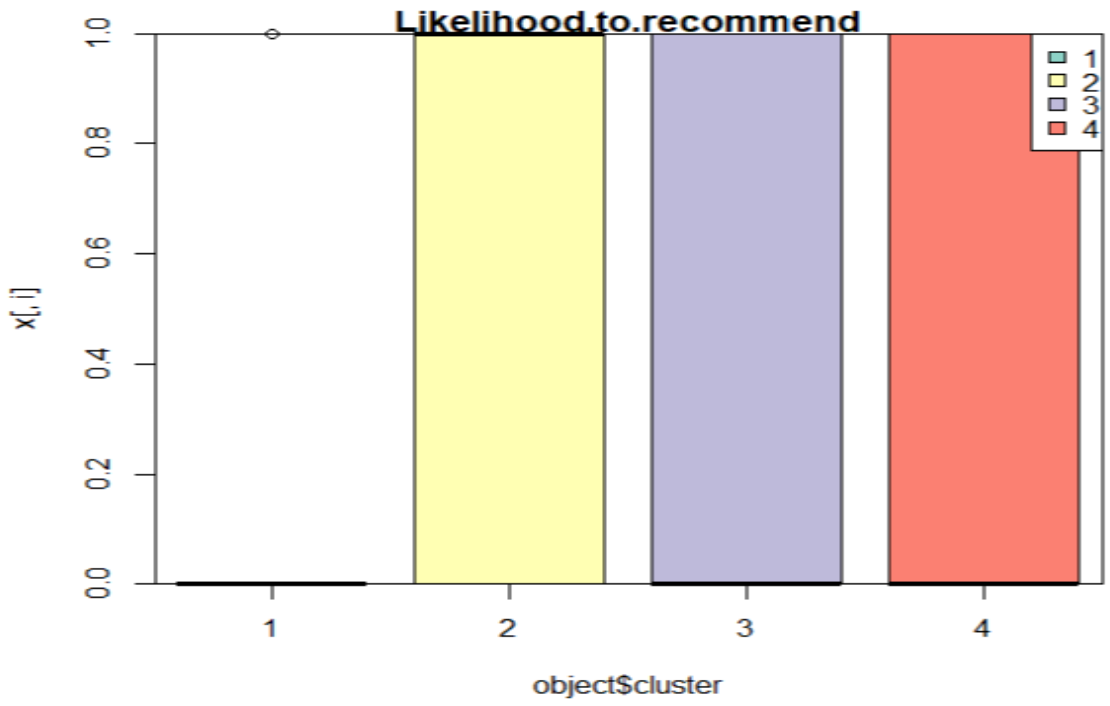
The K-Prototype algorithm accounts for both categorical and continuous variables. It works well with regards to finding "the abnormal clusters" by incorporating all types of attributes into its clusters.

Elbow curve was used to determine the optimal number of clusters categorizing features with relative similar behavior into 5 clusters.



The plots below represent the clusters as per predictors.

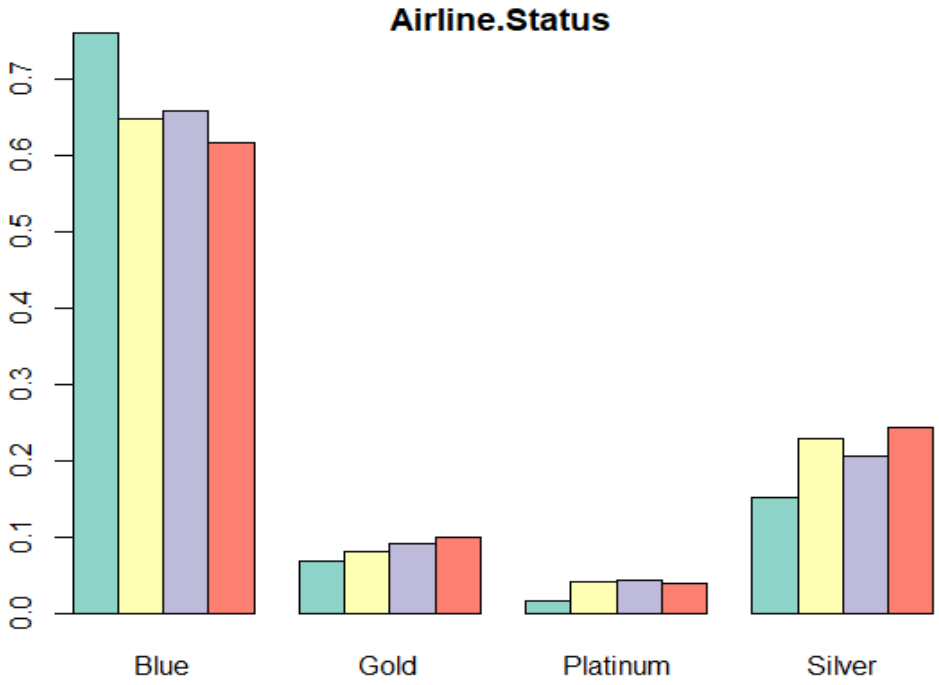
The 5 colors represent the 5 clusters and the bars represent the categorization of Likelihood to Recommend in each of the 4 clusters.
This plot tells us that Cluster 1 is Detractor clusters, i.e.: Likelihood to recommend = 0
and cluster 2,3,4 and 5 are promoter clusters, i.e.: Likelihood to recommend = 1.
We look at the other feature distribution across clusters. If they are significantly populated in clusters 1 and 4, they are detractors, else promoters



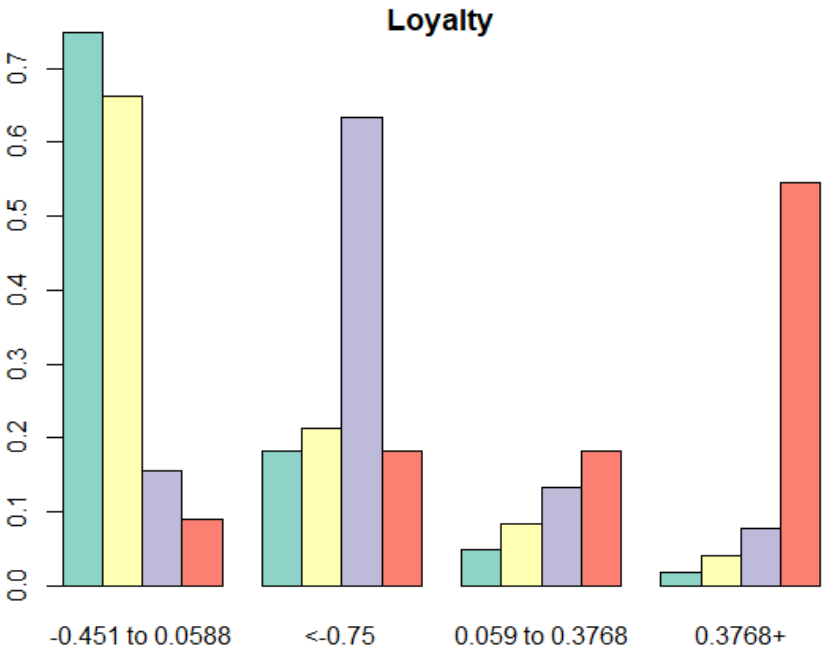
We are classifying the important features extracted by Random Forest, XGBoost to infer the statistical significance of each selected feature using Logistic Regression's beta coefficients.

The below results are supported by the K-Prototype clustering :

The customer whose Airline status is Silver is 12.4% less likely to be a detractor

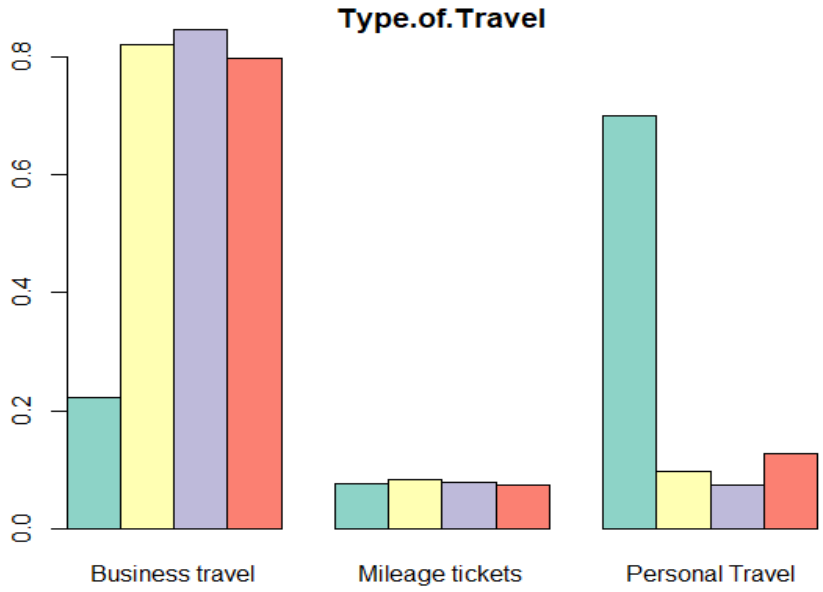


The customer whose Loyalty score is Less than -0.75, is 1.4% less likely to be a detractor.



Customer's with Personal Travel is 21 % more likely to be a detractor.

The customer whose age is greater than 71 years, we predict that the probability of being a detractor increases by 2.8%



Association Rule Mining

Association rule learning is a rule-based machine learning method for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in databases using some measures of interestingness [2].

Based on the previous models, these are the top 10 most variables with high feature importance across all the 3 classifiers:

- Partner Name
- Type of travel
- Age
- Airline Status
- Loyalty
- Total Freq. Flyer. Accts
- Price Sensitivity

- Class
- Departure Delay in Minutes
- Arrival Delay in Minutes

These variables were converted into transactions and applied as input dataset to the apriori algorithm.

```
df_newX <- as(df_new1, "transactions")
unique(df_new1$Likelihood_to_recommend)

inspect(df_newX)
itemFrequency(df_newX)
itemFrequencyPlot(df_newX)

ruleset <- apriori(df_newX,
                   # Specify threshold of 0.005 for support, and 0.5 for confidence. That is, show only those values that are higher than the thresholds.
                   parameter = list(support=0.05, confidence = 0.05, minlen = 3),
                   # Specify the rhs as "Survived = Yes", and all the rest of the variables of the transaction as lhs of the equation
                   appearance = list(default="lhs", rhs = ("Likelihood_to_recommend= 0")))
inspect(ruleset)
inspectDT(ruleset)
```

The results for detractors and promoters are these:

For Detractors:

Characteristics	Values
Type of Travel	Personal Travel
Age	71 years +
	57 to 71 years
Airline Status	Blue
Loyalty	-0.75 to -0.45 (The users generally don't travel by

	this airline)
Flights per Year	26+

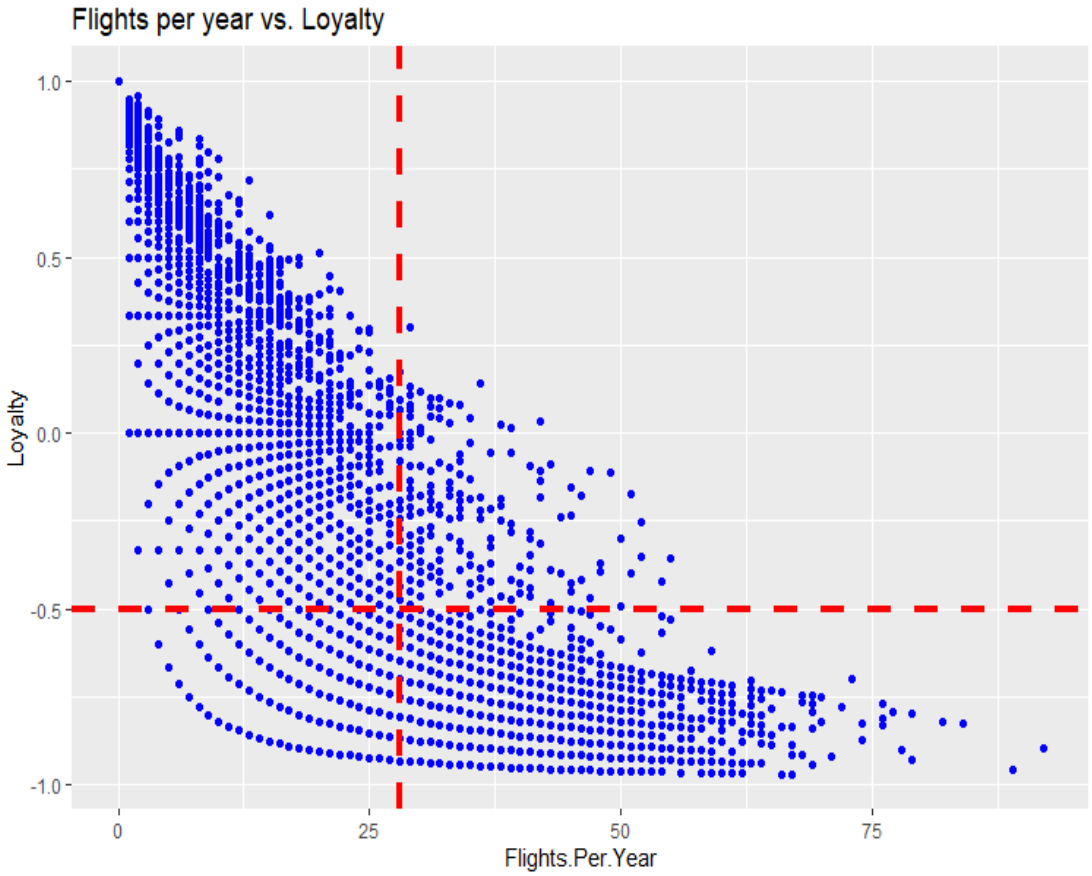
Promoters:

Characteristics	Values
Type of Travel	Business Travel
Departure Delay	Less than 5 min
Arrival Delay	Less than 5 min
Airline Status	Silver
Price Sensitivity	1 (from a range of 0 to 5,i.e. not much sensitive)

RECOMMENDATIONS

Based on the results obtained from the models and the Apriori Algorithms, we suggest some changes or decisions the airline can make to reduce customer churn, and further increase their profits.

1. Frequent Flyers



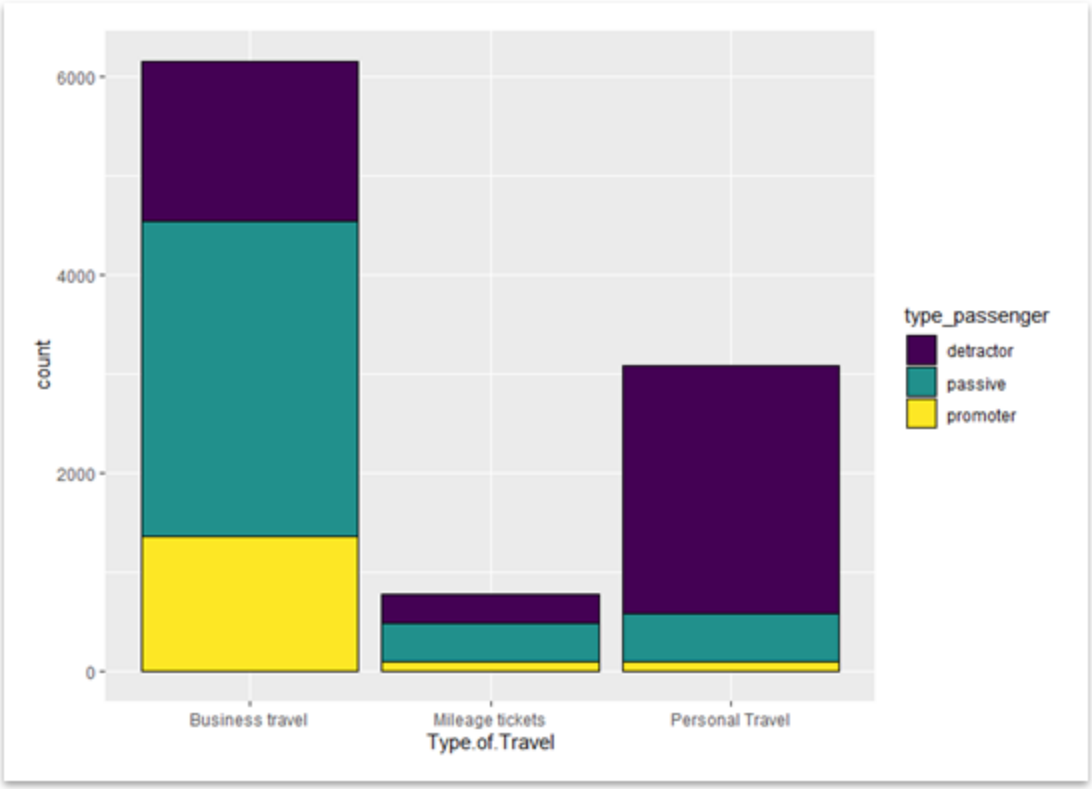
Based on our results, a person flying 28 times or more a year is a detractor, and has relatively low loyalty score, between -0.75 to -0.45. We consider 25% to 30% of those flights are with Southeast, i.e. 6-9+ flights per year are with Southeast. This means that these customers generally don't prefer Southeast Airlines, as they feel that they don't get an optimum offer even after traveling these many flights with Southeast.

Our Advice:

To make the customers return, they must feel that they will receive the most optimum offer if they continue to travel for more than 6-7 flights with Southeast per year. So, we

advise increasing the Mileage points for these customers to offer some sort of discount and also offer complimentary food to these regular flyers.

2. Personal Travelers

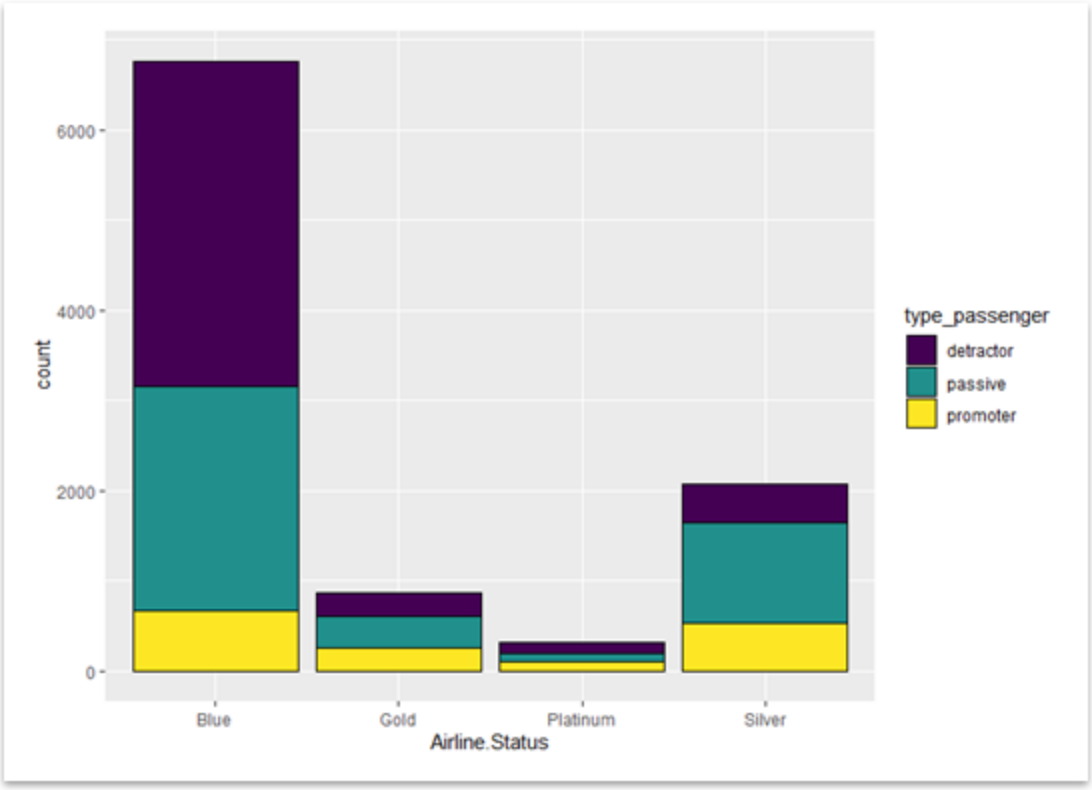


Furthermore, the most dissatisfied customers of the Airline are the customers flying with the intention of Personal Travel. We think that is because the customers pay for their fight themselves, while Business Travel expenses are paid by the Organization, which explains the high rate of Promoters in Business Travels.

Our advice:

Increase efforts to target families, and individual flyers. Advertise family plans, especially during the festive seasons and school holiday seasons to attract families.

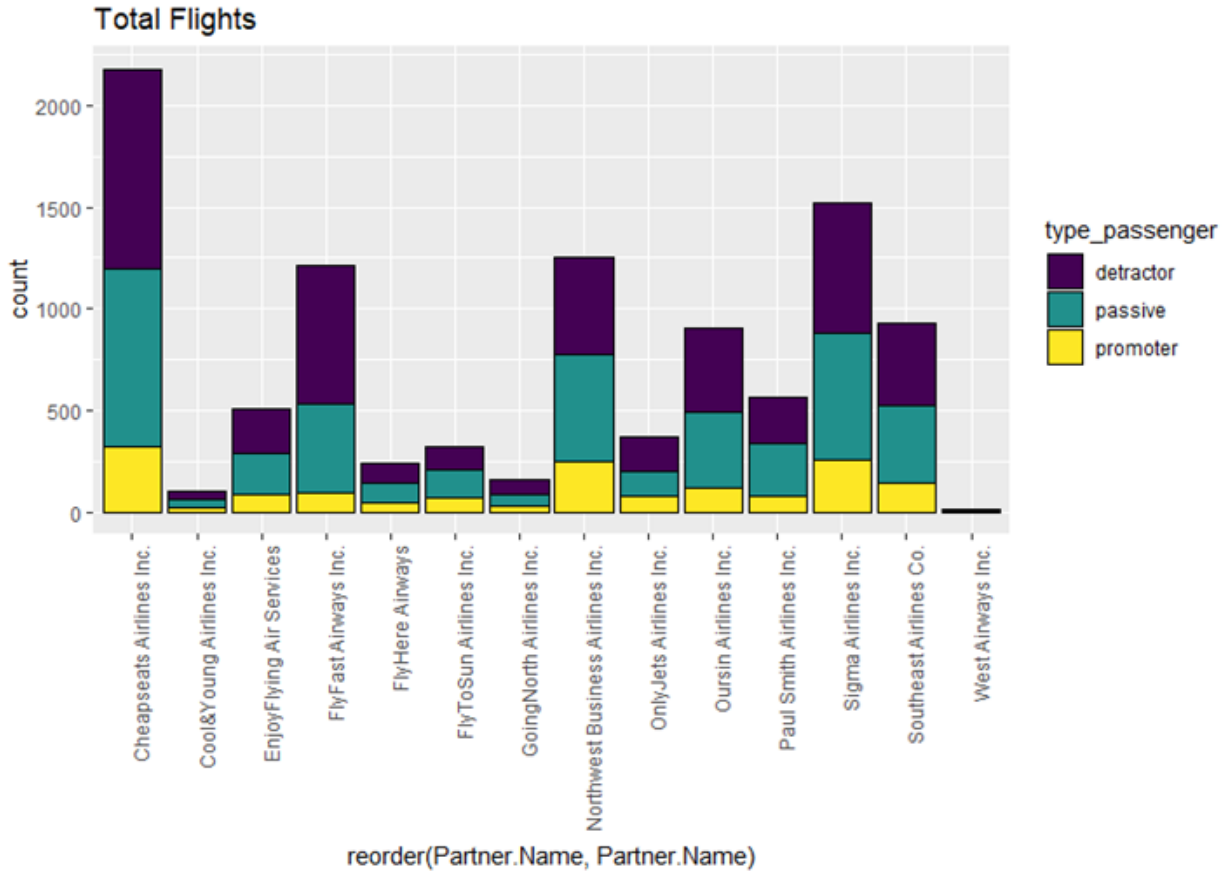
3. Airline status

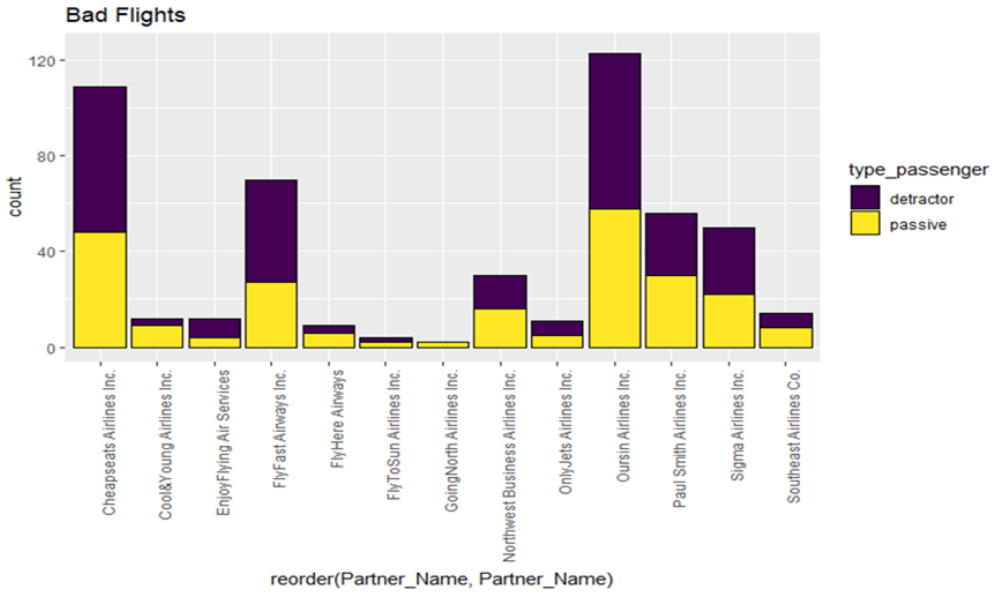


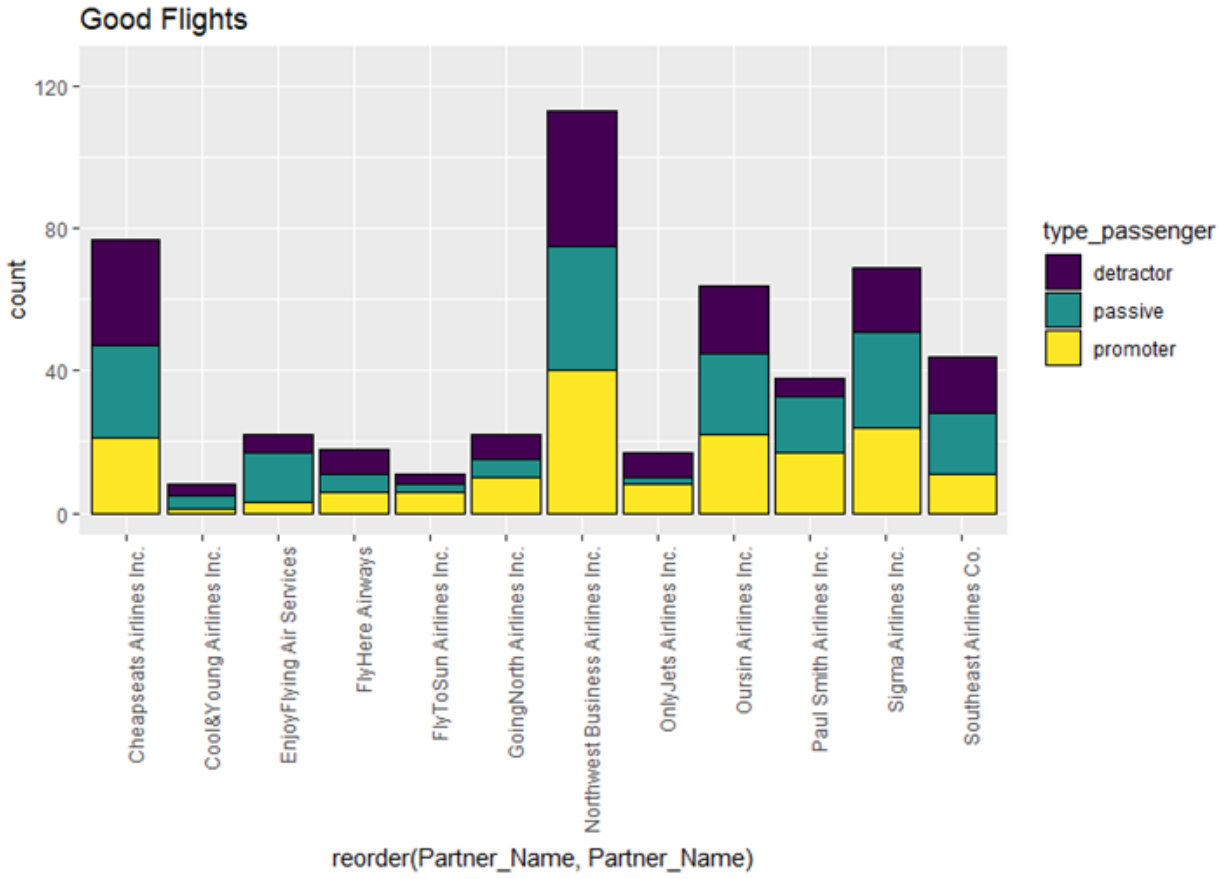
Most promoters have Silver airline status. So, the flights with Status Silver are the best flights that the airline has. And even Gold and Platinum have less proportion of promoters than silver, even without considering the better service provided in them based on price. Also, the detractors are generally from Blue Status flights, as most of the passengers fly through them.

Our advice:
Increase the number of Silver Status flights. At the same time, improve the services of the Blue status flights, as most of the passengers travel by Blue Airline status.

4. Partner Airlines







After analyzing all the unique flights based on their Origin and Destination cities, we calculated the counts and proportions of promoters and detractors for each of them.

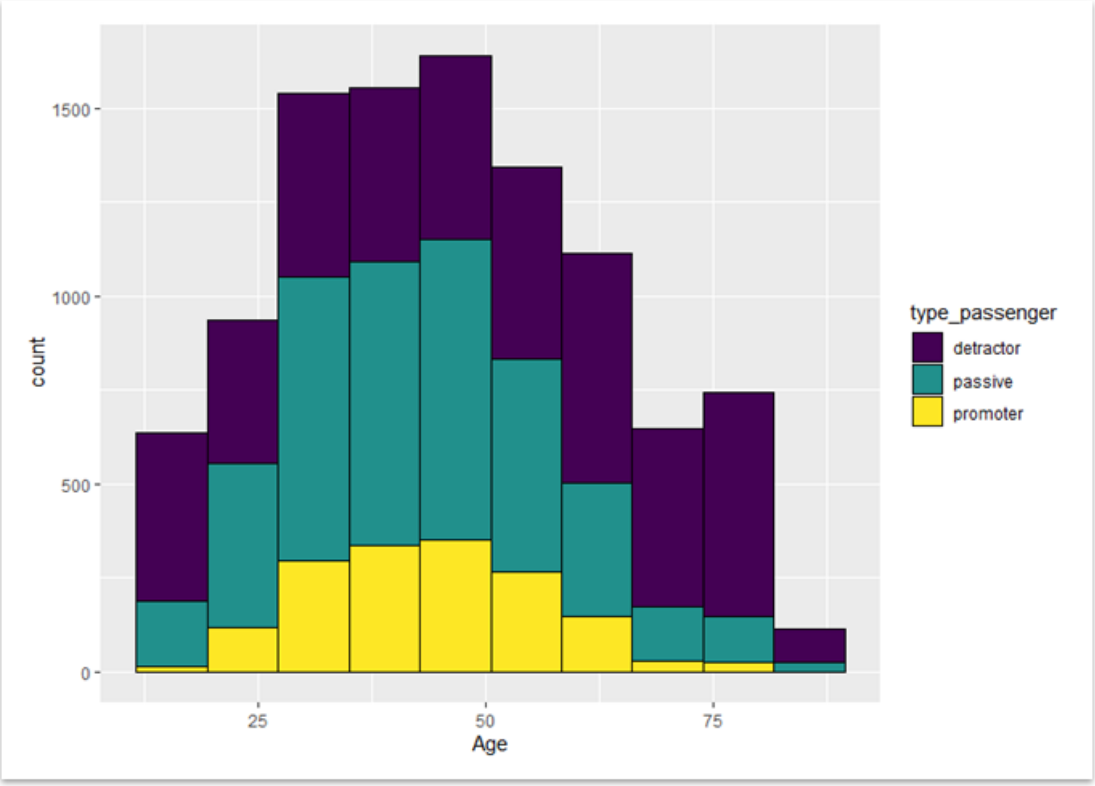
Based on that, we analyzed Partner Airlines to find the worst from them. We analyzed the top 500 worst flights, and the top 500 best flights, and based on those results, our advice is to put the below partner airlines under probation or remove them:

- Cheapseats Airline Inc.
- FlyFast Airways Inc.
- Oursin Airlines Inc.

There is also a special case of Northwest Business Airlines. They provide the most number of Promoters as well as Detractors, so they cannot just be removed or put on probation, as then Southeast airlines capabilities will be severely hampered.

But there is a solution, Audit and ask Northwest Business Airlines to make their services more consistent in services for all their flights. Identify the detractors by the Audit, and make efforts to increase promoters.

5. Better services to senior citizens



Our advice is to provide better services to senior citizens as the number of detractors is more for the age group 57+, so it would be beneficial if the senior citizens are provided with better services like:

- Medical staff/helpers
- Wheelchair assistant
- Special attention to food preferences

REFERENCES

[1] Regression analysis. From Wikipedia the free encyclopedia https://en.wikipedia.org/wiki/Regression_analysis

[2] Association rule learning. From Wikipedia, the free encyclopedia https://en.wikipedia.org/wiki/Association_rule_learning

APPENDIX

```
#IST 687, NPS Project
cat('\014')  # Clear the console
rm(list=ls()) # Clear all user objects from the environment!!!

# Set working directory
# Change to the folder containing your homework data files
setwd("/Users/mistr/OneDrive/Documents/IST687/Homework")

#install.packages("glmnet")
#install.packages("ordinal")
#install.packages("rpart")
#install.packages("randomForest")
#install.packages("caTools")
#install.packages("ROCR")

library(tidyverse)
library(RCurl)
library(jsonlite)
library(imputeTS)
library(ggplot2)
library(ggmap)
library(arules)
library(arulesViz)
library(tm)
library(ordinal)
library(caret)
library(glmnet)
library(rpart)
library(randomForest)
library(caTools)
library(ROCR)

#Import data from data from json file
df_nps <- jsonlite::fromJSON("./fall2019-survey-M09.json")
View(df_nps)

# When we remove na values, the na values mean flight is cancelled. Hence we only get the data for flights which
are not cancelled
df_nps_complete <- df_nps[complete.cases(df_nps[,-32]),]
View(df_nps_complete)

#Find NA values positions in every column
na_values <- apply(apply(df_nps, 2, is.na), 2, which)
na_values
View(na_values)
```

```
# plot the graph displaying all the cities
Destination_plot <- ggplot(df_nps, aes(x = "",y = df_nps$Destination.City, fill = df_nps$Destination.City))+
  geom_bar(stat = "identity")
Destination_plot

# Get the proportion of categories for Airline.Status
prop.table(table(df_nps$Airline.Status))

#Create Recommender type
df_nps_complete$type_passenger <- cut(df_nps_complete$Likelihood.to.recommend,
                      breaks = c(-1,7,9,Inf),
                      labels = c("detractor","passive","promoter"))

glimpse(df_nps_complete)

df_nps_complete$type_passenger <- factor(df_nps_complete$type_passenger, ordered = TRUE)
df_nps_complete$type_passenger[which(is.na(df_nps_complete$type_passenger))] = "passive"
str(df_nps_complete$type_passenger)

# Create Arrival delay > 5 boolean
df_nps_complete$Arrival_Delay_greater_than_5 <- (df_nps_complete$Arrival.Delay.in.Minutes > 5)
str(df_nps_complete$Arrival_Delay_greater_than_5)

# Create Departure delay > 5 boolean
df_nps_complete$Departure_Delay_greater_than_5 <- (df_nps_complete$Departure.Delay.in.Minutes > 5)
str(df_nps_complete$Departure_Delay_greater_than_5)

# Create Long_flight_time
quantile(df_nps_complete$Flight.time.in.minutes)
df_nps_complete$Long_flight_time <- df_nps_complete$Flight.time.in.minutes > 144


# Create Long trip > 1100 boolean
quantile(df_nps_complete$Flight.Distance)
df_nps_complete$Long_Trip <- df_nps_complete$Flight.Distance > 1100

# Create Flight_time
df_nps_complete$flight_day_part <- cut(df_nps_complete$Scheduled.Departure.Hour,
                      breaks = c(-1,6,12,17,19,Inf),
                      labels = c("late night","morning","afternoon","evening","night"))
glimpse(df_nps_complete)


#unique(df_nps$Partner.Code)[order(unique(df_nps$Partner.Code))]

# Create factors for categorical variables for descriptive analysis
df_nps_complete$Gender <- factor(df_nps_complete$Gender, labels = c("Female","Male"))
```

```
df_nps_complete$Airline.Status <- factor(df_nps_complete$Airline.Status, labels = c("Blue", "Gold", "Platinum",
"Silver"))
df_nps_complete$Type.of.Travel <- factor(df_nps_complete$Type.of.Travel, labels = c("Business travel", "Mileage
tickets", "Personal Travel"))
df_nps_complete$Class <- factor(df_nps_complete$Class, labels = c("Business", "Eco", "Eco Plus"))
df_nps_complete$Partner.Code <- factor(df_nps_complete$Partner.Code, labels =
unique(df_nps_complete$Partner.Code)[order(unique(df_nps_complete$Partner.Code))])
df_nps_complete$Partner.Name<- factor(df_nps_complete$Partner.Name, labels =
unique(df_nps_complete$Partner.Name)[order(unique(df_nps_complete$Partner.Name))])
df_nps_complete$Origin.City<- factor(df_nps_complete$Origin.City, labels =
unique(df_nps_complete$Origin.City)[order(unique(df_nps_complete$Origin.City))])
df_nps_complete$Origin.State<- factor(df_nps_complete$Origin.State, labels =
unique(df_nps_complete$Origin.State)[order(unique(df_nps_complete$Origin.State))])
df_nps_complete$Destination.City<- factor(df_nps_complete$Destination.City, labels =
unique(df_nps_complete$Destination.City)[order(unique(df_nps_complete$Destination.City))])
df_nps_complete$Destination.State<- factor(df_nps_complete$Destination.State, labels =
unique(df_nps_complete$Destination.State)[order(unique(df_nps_complete$Destination.State))])

# Create train and test dataset
set.seed(101)
sample = sample.split(df_nps_complete$Likelihood.to.recommend, SplitRatio = .75)
train_df = subset(df_nps_complete, sample == TRUE)
test_df  = subset(df_nps_complete, sample == FALSE)
glimpse(train_df)
glimpse(test_df)
#View(df_nps_complete)
str(df_nps_complete)

# write the file to csv
write_csv(df_nps_complete, "airline_data.csv")
View(df_nps_complete)

# PLOTS

library(sqldf)
library(ggmap)

# get all the unique flights
airports <- sqldf(
   "
   SELECT d1.Origin.City, d1.Destination.City, d1.olong, d1.olat, d1.dlong, d1.dlat
   FROM df_nps_complete d1
   GROUP BY d1.Origin.City, d1.Destination.City, d1.olong, d1.olat, d1.dlong, d1.dlat
   "
)

# histogram
ggplot(df_nps_complete, aes(x = type_passenger))+
```

```
  geom_histogram(stat = "count", fill = 'light Blue', color = "Black")

prop.table(table(df_nps_complete$type_passenger))

ggplot(df_nps_complete, aes(x = "",y = "", fill = type_passenger))+
  geom_bar(stat = "identity", width = 1)+
  coord_polar("y", start = 0)

# Apriori Transactions
glimpse(df_nps_complete)
dim(df_nps_complete)
df_nps_complete1 = df_nps_complete[,c(1,2,3,7,8,9,10,14,15,16,17,19,20,33,34,35,36,37,38)]

apply(apply(df_nps_complete1, 2, is.na), 2, which)
df_nps_complete1$type_passenger[which(is.na(df_nps_complete1$type_passenger))] = "passive"

df_nps_completeX <- as(df_nps_complete1, "transactions")

inspect(df_nps_completeX)
itemFrequency(df_nps_completeX)
itemFrequencyPlot(df_nps_completeX)

ruleset <- apriori(df_nps_completeX,
                   # Specify threshold of 0.005 for support, and 0.5 for confidence. That is, show only those values that are
higher than the thresholds.
                   parameter = list(support=0.05, confidence = 0.05, minlen = 3),
                   # Specify the rhs as "Survived = Yes", and all the rest of the variables of the transaction as lhs of the
equation
                   appearance = list(default="lhs", rhs = ("type_passenger=passive")))
inspect(ruleset)
inspectDT(ruleset)

#Promoter rules
#{Type.of.Travel=Business travel,Class=Eco}            {type_passenger=promoter}           0.110     0.222     1.441
1,132.000
#{Gender=Male,Type.of.Travel=Business travel,Class=Eco}    {type_passenger=promoter}           0.065     0.269
1.747     670.000
#{Gender=Male,Type.of.Travel=Business travel}       {type_passenger=promoter}           0.076     0.267     1.734
786.000
#{Type.of.Travel=Business travel,flight_day_part=morning}    {type_passenger=promoter}           0.052     0.225
1.463     537.000
#{Type.of.Travel=Business travel,Class=Eco}            {type_passenger=promoter}           0.110     0.222     1.441
1,132.000
#{Gender=Male,Class=Eco}           {type_passenger=promoter}           0.074     0.200     1.300     757.000

#Detractor
# {Airline.Status=Blue,Type.of.Travel=Personal Travel}          {type_passenger=detractor}          0.213     0.886
1.997     2,186.000
```

```
# {Airline.Status=Blue,Type.of.Travel=Personal Travel,Class=Eco}      {type_passenger=detractor}          0.174
0.885     1.995     1,791.000
# {Gender=Female,Type.of.Travel=Personal Travel,Class=Eco,Arrival_Delay_greater_than_5}
{type_passenger=detractor}          0.055     0.928     2.091     565.000
# {Type.of.Travel=Personal Travel,Class=Eco,Arrival_Delay_greater_than_5}    {type_passenger=detractor}
0.084     0.921     2.076     859.000
# {Type.of.Travel=Personal Travel,Arrival_Delay_greater_than_5}       {type_passenger=detractor}          0.100
0.920     2.075     1,028.000
# {Airline.Status=Blue,Type.of.Travel=Personal Travel,flight_day_part=afternoon}
{type_passenger=detractor}          0.072     0.903     2.036     738.000
# {Airline.Status=Blue,Type.of.Travel=Personal Travel,Class=Eco,flight_day_part=afternoon}
{type_passenger=detractor}          0.059     0.902     2.034     610.000
# {Airline.Status=Blue,Gender=Female,Type.of.Travel=Personal Travel}          {type_passenger=detractor}
0.140     0.897     2.021     1,441.000
```

PLOTS

```
boxplot(df_nps_complete$Age ~ df_nps_complete$type_passenger)

df_nps_complete %>%
  select(Price.Sensitivity, type_passenger) %>%
  group_by(type_passenger) %>%
  ggplot(aes(x = Price.Sensitivity, y = type_passenger))+
    geom_jitter(alpha = 0.4, color = 'light green')+
      theme(
        panel.background = element_rect(fill = "black",
                                        colour = "black",
                                        size = 0.5, linetype = "solid"),
        panel.grid.major = element_line(size = 0.5, linetype = 'solid',
                                        colour = "orange"),
        panel.grid.minor = element_line(size = 0.25, linetype = 'solid',
                                        colour = "orange")
  )

# tpye_passenger vs Gender

df_nps_complete %>%
  select(Gender, type_passenger) %>%
  group_by(type_passenger) %>%
  ggplot(aes(x = Gender, y = type_passenger))+
    geom_jitter(alpha = 0.4, color = 'light green')+
      theme(
        panel.background = element_rect(fill = "black",
                                        colour = "black",
                                        size = 0.5, linetype = "solid"),
        panel.grid.major = element_line(size = 0.5, linetype = 'solid',
                                        colour = "orange"),
        panel.grid.minor = element_line(size = 0.25, linetype = 'solid',
                                        colour = "orange")
```

```
  )

# type_passenger vs Type.of.Travel

df_nps_complete %>%
  select(Type.of.Travel, type_passenger) %>%
  group_by(type_passenger) %>%
  ggplot(aes(x = Type.of.Travel, y = type_passenger))+
    geom_jitter(alpha = 0.4, color = 'light green')+
      theme(
        panel.background = element_rect(fill = "black",
                                        colour = "black",
                                        size = 0.5, linetype = "solid"),
        panel.grid.major = element_line(size = 0.5, linetype = 'solid',
                                        colour = "orange"),
        panel.grid.minor = element_line(size = 0.25, linetype = 'solid',
                                        colour = "orange")
  )

# type_passenger vs Airline.Status

df_nps_complete %>%
  select(Airline.Status, type_passenger) %>%
  group_by(type_passenger) %>%
  ggplot(aes(x = Airline.Status, y = type_passenger))+
  geom_jitter(alpha = 0.4, color = 'light green')+
  theme(
    panel.background = element_rect(fill = "black",
                                    colour = "black",
                                    size = 0.5, linetype = "solid"),
    panel.grid.major = element_line(size = 0.5, linetype = 'solid',
                                    colour = "orange"),
    panel.grid.minor = element_line(size = 0.25, linetype = 'solid',
                                    colour = "orange")
  )


# type_passenger vs Class

df_nps_complete %>%
  select(Class, type_passenger) %>%
  group_by(type_passenger) %>%
  ggplot(aes(x = Class, y = type_passenger))+
  geom_jitter(alpha = 0.4, color = 'light green')+
  theme(
    panel.background = element_rect(fill = "black",
                                    colour = "black",
                                    size = 0.5, linetype = "solid"),
```

```
    panel.grid.major = element_line(size = 0.5, linetype = 'solid',
                                    colour = "orange"),
    panel.grid.minor = element_line(size = 0.25, linetype = 'solid',
                                    colour = "orange")
  )

ggplot(df_nps_complete, aes(Age))+
geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black')

ggplot(df_nps_complete, aes(Arrival.Delay.in.Minutes))+
geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black')

ggplot(df_nps_complete, aes(Flight.cancelled))+
geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black', stat = "count")

ggplot(df_nps_complete, aes(Flight.Distance))+
 geom_histogram(aes(fill = type_passenger), bins = 15, color = 'Black')


ggplot(df_nps_complete[df_nps_complete$Loyalty<-0.75,], aes(x = "",y = "", fill = type_passenger))+
geom_bar(stat = "identity", width = 1)+
 coord_polar("y", start = 0)

ggplot(df_nps_complete[df_nps_complete$Loyalty>=-0.75,], aes(x = "",y = "", fill = type_passenger))+
  geom_bar(stat = "identity", width = 1)+
  coord_polar("y", start = 0)

ggplot(df_nps_complete, aes(Loyalty))+
  geom_histogram(aes(fill = type_passenger), bins = 5, color = 'Black')

ggplot(df_nps_complete, aes(Shopping.Amount.at.Airport))+
  geom_histogram(aes(fill = type_passenger), bins = 15, color = 'Black')

ggplot(df_nps_complete, aes(Eating.and.Drinking.at.Airport))+
  geom_histogram(aes(fill = type_passenger), bins = 15, color = 'Black')

ggplot(df_nps_complete, aes(Eating.and.Drinking.at.Airport))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black')

ggplot(df_nps_complete, aes(Age))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black')

ggplot(df_nps_complete, aes(Class))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black', stat = "count")


ggplot(df_nps_complete, aes(Age))+
  geom_histogram(aes(fill = Gender), bins = 10, color = 'Black')
```

```
ggplot(df_nps_complete, aes(Price.Sensitivity))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black')

ggplot(df_nps_complete, aes(Flights.Per.Year))+
  geom_histogram(aes(fill = type_passenger), bins = 20, color = 'Black')

ggplot(df_nps_complete, aes(Total.Freq.Flyer.Accts))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black')

ggplot(df_nps_complete, aes(Day.of.Month))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black')

ggplot(df_nps_complete, aes(Flight.time.in.minutes))+
  geom_histogram(aes(fill = type_passenger), bins = 15, color = 'Black')

ggplot(df_nps_complete, aes(Year.of.First.Flight))+
  geom_histogram(aes(fill = type_passenger),binwidth = 1, bins = 10, color = 'Black')

ggplot(df_nps_complete, aes(Type.of.Travel))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black', stat = 'count')

ggplot(df_nps_complete, aes(flight_day_part))+
 geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black', stat = "count")

ggplot(df_nps_complete, aes(flight_day_part))+
 geom_histogram(aes(fill = Arrival_Delay_greater_than_5), bins = 10, color = 'Black', stat = "count")

ggplot(df_nps_complete, aes(Partner.Code))+
 geom_histogram(aes(fill = Arrival_Delay_greater_than_5), bins = 10, color = 'Black', stat = "count")

ggplot(df_nps_complete[df_nps_complete$Partner.Name == 'Northwest Business Airlines Inc.',], aes(x="", y="", fill
= type_passenger))+
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y", start = 0)




# Loyalty vs Recommender
df_nps_complete$Loyalty_type <- ifelse(df_nps_complete$Loyalty  < 0, 'below', 'above')
ggplot(df_nps_complete, aes(x = Likelihood.to.recommend, y = Loyalty_type, label = Loyalty))+
geom_bar(stat = "identity", aes(fill=Loyalty_type), width=.5)  +
 scale_fill_manual(name="Loyalty",
                   labels = c("Above 0", "Below 0"),
                   values = c("above"="#00ba38", "below"="#f8766d"))

# Average delay vs Recommender
```

```
ggplot(df_nps_complete, aes(x = Likelihood.to.recommend, y = Arrival_Delay_greater_than_5, label = Loyalty))+
  geom_bar(stat = "identity", aes(fill=Loyalty_type), width=.5) +
  scale_fill_manual(name="Arrival Delay",
                    labels = c(">5 min", "<=5 min"),
                    values = c("#f8766d","#00ba38"))

# Eating and Drinking
prop.table(table(df_nps_complete$type_passenger, df_nps_complete$Eating.and.Drinking.at.Airport))



# map plt
us <- map_data("state")
df_nps_complete$Origin.State <- tolower(df_nps_complete$Origin.State)
df_nps_complete$Destination.State <- tolower(df_nps_complete$Destination.State)

flight_state_destination <- ggplot(df_nps_complete, aes(map_id = Destination.State))+
  geom_map(map = us, aes(fill = Arrival_Delay_greater_than_5), color = 'Black')+
  expand_limits(x = us$long, y = us$lat)+
  #geom_point(aes(x = dlong, y = dlat))+
  ggtitle("Destination State")+
  coord_map()
flight_state_destination

flight_state_origin <- ggplot(df_nps_complete, aes(map_id = Origin.State))+
  geom_map(map = us, aes(fill = Departure_Delay_greater_than_5), color = 'Black')+
  expand_limits(x = us$long, y = us$lat)+
  #geom_point(aes(x = olong, y = olat))+
  ggtitle("Origin State")+
  coord_map()
flight_state_origin



# Find important words from freeText
which(!is.na(df_nps_complete$freeText))
comments <- df_nps_complete$freeText[which(!is.na(df_nps_complete$freeText))]

str(comments)
View(comments)
comments.vec <- VectorSource(comments)
comments.corpus <- Corpus(comments.vec)

comments.corpus <- tm_map(comments.corpus, content_transformer(tolower))
comments.corpus <- tm_map(comments.corpus, removePunctuation)
comments.corpus <- tm_map(comments.corpus, removeNumbers)
comments.corpus <- tm_map(comments.corpus, removeWords, stopwords("english"))

inspect(comments.corpus)
```

```
#as(comments.corpus, "transactions")

tdm <- TermDocumentMatrix(comments.corpus)

inspect(tdm)

wordCounts <- rowSums(as.matrix(tdm))
wordCounts <- sort(wordCounts, decreasing = TRUE)

View(wordCounts)

#df_nps_complete$Gender <- factor(df_nps_complete$Gender, labels = "")
#df_nps_complete$Airline.Status <- factor(df_nps_complete$Airline.Status, labels = "")
#df_nps_complete$Type.of.Travel <- factor(df_nps_complete$Type.of.Travel, labels = "")
#df_nps_complete$Class <- factor(df_nps_complete$Class, labels = "")
#df_nps_complete$Partner.Code <- factor(df_nps_complete$Partner.Code, labels = "")
#df_nps_complete$Partner.Name<- factor(df_nps_complete$Partner.Name, labels = "")
#df_nps_complete$Origin.City<- factor(df_nps_complete$Origin.City, labels = "")
#df_nps_complete$Origin.State<- factor(df_nps_complete$Origin.State, labels = "")
#df_nps_complete$Destination.City<- factor(df_nps_complete$Destination.City, labels = "")
#df_nps_complete$Destination.State<- factor(df_nps_complete$Destination.State, labels = "")

# binning data Apriori
rm(list=ls())
df_nps <- jsonlite::fromJSON("./fall2019-survey-M09.json")

df_new<-read.csv("data_new2.csv")
df_new<-df_new[,-1]
glimpse(df_new)
dim(df_new)
df_new <- data.frame(apply(df_new,2,as.factor))
df_new1 = df_new[,c(1,4,5,7,8,9,10,11,12,13,14,15,16)]
glimpse(df_new1)

gsub("\\.","_",names(df_new1))
names(df_new1) <- gsub("\\.","_",names(df_new1))

apply(apply(df_new1, 2, is.na), 2, which)
df_new1$type_passenger[which(is.na(df_new1$type_passenger))] = "passive"

df_newX <- as(df_new1, "transactions")
unique(df_new1$Likelihood_to_recommend)

inspect(df_newX)
itemFrequency(df_newX)
itemFrequencyPlot(df_newX)
```

```
ruleset <- apriori(df_newX,
            # Specify threshold of 0.005 for support, and 0.5 for confidence. That is, show only those values that are
higher than the thresholds.
            parameter = list(support=0.05, confidence = 0.05, minlen = 3),
            # Specify the rhs as "Survived = Yes", and all the rest of the variables of the transaction as lhs of the
equation
            appearance = list(default="lhs", rhs = ("Likelihood_to_recommend= 0")))
inspect(ruleset)
inspectDT(ruleset)
```

Type of travel: Personal travel

```
ggplot(df_nps_complete, aes(Airline.Status))+
  geom_histogram(aes(fill = type_passenger), bins = 10, binwidth = 0.5, color = 'Black', stat = 'count') +
  theme(axis.text.x = element_text(angle = 0, hjust = 1))


# type_passenger in df_new
dim(df_new)
dim(df_nps)
df_new$type_passenger <- cut(df_nps$Likelihood.to.recommend,
                        breaks = c(-1,7,9,Inf),
                        labels = c("detractor","passive","promoter"))

table(df_new$type_passenger)
df_new$type_passenger[which(is.na(df_new$type_passenger))] = "passive"
df_new$type_passenger <- factor(df_new$type_passenger, ordered = TRUE)

names(df_new) <- gsub("\\.","_",names(df_new))

df_nps$type_passenger <- cut(df_nps$Likelihood.to.recommend,
                         breaks = c(-1,7,9,Inf),
                         labels = c("detractor","passive","promoter"))

table(df_nps$type_passenger)
df_nps$type_passenger[which(is.na(df_nps$type_passenger))] = "passive"
df_nps$type_passenger <- factor(df_nps$type_passenger, ordered = TRUE)


df_nps1 <- copy(df_nps)
names(df_nps1) <- gsub("\\.","_",names(df_nps1))

df_nps1$promoter <- ifelse(df_nps1$type_passenger == 'promoter',1,0)
df_nps1$detractor <- ifelse(df_nps1$type_passenger == 'detractor',1,0)
df_nps1$passive <- ifelse(df_nps1$type_passenger == 'passive',1,0)
df_nps1$not_promoter <- df_nps1$passive+df_nps1$detractor

glimpse(df_nps1)
```

```
library(sqldf)
df_nps2 <- sqldf("
            SELECT d1.Origin_City, d1.Destination_City, d1.Origin_State, d1.Destination_State, d1.olong, d1.olat,
d1.dlong, d1.dlat, COUNT(olong) As Count, SUM(promoter) as promoter, SUM(not_promoter) as not_promoter,
SUM(detractor) as detractor, SUM(passive) as passive, SUM(promoter)/COUNT(olong) as promoter_per,
SUM(not_promoter)/COUNT(olong) as not_promoter_per
            FROM df_nps1 AS d1
            GROUP BY  d1.Origin_City, d1.Destination_City, d1.Origin_State, d1.Destination_State, d1.olong,
d1.olat, d1.dlong, d1.dlat")

View(df_nps2)

df_nps2_busy_origin <- sqldf("
                SELECT d1.Origin_City, COUNT(d1.Destination_City) as busy
                FROM df_nps2 as d1
                GROUP BY d1.Origin_City
                ")

View(df_nps2_busy_origin)

df_busy <- sqldf("
            SELECT d1.*
            FROM df_nps1 AS d1 INNER JOIN df_nps2_busy_origin d2 ON d1.Origin_City = d2.Origin_City
            WHERE d2.busy > 70
            ")
View(df_busy)

ggplot(df_busy, aes(Departure_Delay_in_Minutes))+
 geom_histogram(aes(fill = type_passenger),bins = 10, color ='black')

ggplot(df_nps, aes(type_passenger))+
  geom_histogram(stat = 'count')


df_good <- sqldf("
            SELECT *
            FROM df_nps1 d1
            WHERE d1.Origin_City = 'Buffalo, NY' and d1.Destination_City = 'New York, NY'")

View(df_good)

df_good2 <- sqldf("
            SELECT *
            FROM df_nps1 d1
            WHERE d1.Origin_City = 'San Diego, CA' and d1.Destination_City = 'Los Angeles, CA'")

View(df_good2)
```

```
df_good3 <- sqldf("
            SELECT d1.*
            FROM df_nps1 d1 LEFT JOIN df_nps2 d2 ON d1.Origin_City = d2.Origin_City and d1.Destination_City
= d2.Destination_City
            WHERE d2.promoter_per > 0.27 and d2.Count > 10")

View(df_good3)

df_bad <- sqldf("
            SELECT *
            FROM df_nps1 d1
            WHERE d1.Origin_City = 'Houston, TX'")

View(df_bad)

df_bad2 <- sqldf("
            SELECT d1.*
            FROM df_nps1 d1 LEFT JOIN df_nps2 d2 ON d1.Origin_City = d2.Origin_City and d1.Destination_City
= d2.Destination_City
            WHERE d2.not_promoter_per = 1 and d2.Count > 10")

View(df_bad2)

#plots for comparison

dim(df_bad2)
dim(df_good3)
ggplot(df_bad2, aes(Age))+
  geom_histogram(aes(fill = Airline_Status), bins = 10, color = 'Black') +
  ggtitle("Bad Flights")+
  theme(axis.text.x = element_text(angle = 0))

ggplot(df_good3, aes(Age))+
  geom_histogram(aes(fill = Airline_Status), bins = 10, color = 'Black') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 0))

# class
ggplot(df_bad2, aes(Class))+
  geom_histogram(aes(fill = Airline_Status), bins = 10, color = 'Black', stat = 'count') +
  ggtitle("Bad Flights")+
  theme(axis.text.x = element_text(angle = 0))

ggplot(df_good3, aes(Class))+
  geom_histogram(aes(fill = Airline_Status), bins = 10, color = 'Black', stat = 'count') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 0))
```

```
# Type of travel
ggplot(df_bad2, aes(Type_of_Travel))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black', stat = 'count') +
  ggtitle("Bad Flights")+
  theme(axis.text.x = element_text(angle = 0))+ylim(0, 400)

ggplot(df_good3, aes(Type_of_Travel))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black', stat = 'count') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 0))+ylim(0, 400)

# Flight time
ggplot(df_bad2, aes(Flight_time_in_minutes))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black') +
  ggtitle("Bad Flights")+
  theme(axis.text.x = element_text(angle = 0))+ylim(0,200)

ggplot(df_good3, aes(Flight_time_in_minutes))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 0))+ylim(0,200)

#flight distance
ggplot(df_bad2, aes(Flight_Distance))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black') +
  ggtitle("Bad Flights")+
  theme(axis.text.x = element_text(angle = 0))

ggplot(df_good3, aes(Flight_Distance))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 0))

#Partner.Name
ggplot(df_bad2, aes(reorder(Partner_Name,Partner_Name)))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black', stat = 'count') +
  ggtitle("Bad Flights")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+ylim(0,125)

ggplot(df_good3, aes(reorder(Partner_Name,Partner_Name)))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black', stat = 'count') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 90, hjust=1))+ylim(0,125)

ggplot(df_nps, aes(reorder(Partner.Name,Partner.Name)))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black', stat = 'count') +
  ggtitle("Total Flights")+
```

```
  theme(axis.text.x = element_text(angle = 90, hjust=1))


#Flights per year
ggplot(df_nps, aes(x = Flights.Per.Year, y = Loyalty))+
  geom_point(color = "blue")+geom_vline(xintercept = 28, linetype = 'dashed', color = 'red', size = 1.5)+
  geom_hline(yintercept = -0.5, linetype = 'dashed', color = 'red', size = 1.5)+
  ggtitle("Flights per year vs. Loyalty")

ggplot(df_nps[df_nps$Type.of.Travel == 'Mileage tickets',], aes(Flights.Per.Year))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 90, hjust=1))

ggplot(df_good3, aes(Flight_Distance))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 0))

# Type of travel
ggplot(df_nps, aes(Type.of.Travel))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black', stat = 'count') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 0))


ggplot(df_nps, aes(Age))+
  geom_histogram(aes(fill = Type.of.Travel), bins = 10, color = 'Black') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 0))

df_nps_complete %>%
  select(Type.of.Travel, type_passenger) %>%
  group_by(type_passenger) %>%
  ggplot(aes(x = Type.of.Travel, y = type_passenger))+
  geom_jitter(alpha = 0.4, color = 'light green')+
  theme(
    panel.background = element_rect(fill = "black",
                                    colour = "black",
                                    size = 0.5, linetype = "solid"),
    panel.grid.major = element_line(size = 0.5, linetype = 'solid',
                                    colour = "orange"),
    panel.grid.minor = element_line(size = 0.25, linetype = 'solid',
                                    colour = "orange")
  )
```

```
# map plt
us <- map_data("state")
df_nps2$Origin_State <- tolower(df_nps2$Origin_State)
df_nps2$Destination_State <- tolower(df_nps2$Destination_State)

flight_state_destination <- ggplot(df_nps2, aes(map_id = Origin_State))+
  geom_map(map = us, aes(fill = recommender_type), color = 'Black')+
  expand_limits(x = us$long, y = us$lat)+
  #geom_point(aes(x = dlong, y = dlat))+
  ggtitle("Destination State")+
  coord_map()
flight_state_destination

glimpse(df_nps)
# flight map
#install.packages("ggrepel")
library(ggrepel)
#worldmap <- borders("world", colour="#efede1", fill="#efede1") # create a layer of borders

df_nps$type_passenger <- cut(df_nps$Likelihood.to.recommend,
                             breaks = c(-1,7,9,Inf),
                             labels = c("detractor","passive","promoter"))

df_nps$type_passenger <- factor(df_nps$type_passenger, ordered = TRUE)
df_nps$type_passenger[which(is.na(df_nps$type_passenger))] = "passive"



#ggplot() + worldmap +
df_nps$Origin_State <- tolower(df_nps$Origin.State)
df_nps$Destination_State <- tolower(df_nps$Destination.State)
ggplot(df_nps, aes(map_id = Origin_State))+
  geom_map(map = us, color = 'Black')+
  expand_limits(x = us$long, y = us$lat)+
  geom_curve(data=df_nps[df_nps$type_passenger == 'promoter',], aes(x = olong, y = olat, xend = dlong, yend =
dlat, color = factor(type_passenger)), size = 0.2, curvature = .2)

  #geom_point(data=airports, aes(x = lon, y = lat), col = "#970027")
  #geom_text_repel(data=df_nps, aes(x = lon, y = lat, label = airport), col = "black", size = 2, segment.color = NA) +
  theme(panel.background = element_rect(fill="white"),
        axis.line = element_blank(),
        axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        axis.title.x = element_blank(),
        axis.title.y = element_blank()
  )
```

```
library(RCurl)
library(jsonlite)

dataset <-
getURL("https://s3.us-east-1.amazonaws.com/blackboard.learn.xythos.prod/5956621d575cd/8614406?response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27fall2019-survey-M02%25281%2529.json&response-content-type=application%2Fjson&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20191203T035139Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=AKIAIL7WQYDOOHAZJGWQ%2F20191203%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=013be809c46067b40d4620771ebe272898fc8764bf374a90bed830f13a50fab2")
df <- jsonlite::fromJSON(dataset)
write.csv(df,"E:\\Disk partition\\Syracuse ADS\\Syracuse ADS\\1st Semester\\IST 687 Introduction to Data Science\\project\\airline_customer_churner\\data.csv")
#reading raw data from csv
df <- jsonlite::fromJSON('E:/Disk partition/Syracuse ADS/Syracuse ADS/1st Semester/IST 687 Introduction to Data Science/project/fall2019-survey-M07.json')

View(df)
#coverting features into factors and numeric for categorical inference
df$Eating.and.Drinking.at.Airport<-as.numeric(df$Eating.and.Drinking.at.Airport)
df$Day.of.Month<-as.numeric(df$Day.of.Month)
df$Partner.Code<-as.factor(df$Partner.Code)
df$Partner.Name<-as.factor(df$Partner.Name)
df$Airline.Status<-as.factor(df$Airline.Status)
df$Type.of.Travel<-as.factor(df$Type.of.Travel)
df$Gender<-as.factor(df$Gender)
df$Flight.cancelled<-as.factor(df$Flight.cancelled)
df$Price.Sensitivity<-as.factor(df$Price.Sensitivity)
df$Class<-as.factor(df$Class)
df$Likelihood.to.recommend[(df$Likelihood.to.recommend<7)] <- 0 # 'Detractor'
df$Likelihood.to.recommend[(df$Likelihood.to.recommend==7)] <- 0 #'Passive'
df$Likelihood.to.recommend[(df$Likelihood.to.recommend==8)] <- 0# 'Passive'
df$Likelihood.to.recommend[(df$Likelihood.to.recommend==9)] <- 1# 'Promoter'
df$Likelihood.to.recommend[(df$Likelihood.to.recommend==10)] <-1# 'Promoter'
df$Likelihood.to.recommend<-as.factor(df$Likelihood.to.recommend)

#Feature Engineering : bucketing features in categories
quantile(df$Age,c(0,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1))
df$Age<-  ifelse(df$Age<=24,'<24',
                 ifelse(df$Age<=35,'25-35',
                        ifelse(df$Age<=46,'36-46',
                               ifelse(df$Age<=58,'47-58',
                                      ifelse(df$Age<=71,'57-71','71+')))))
table(df$Age)

quantile(df$Flights.Per.Year,c(0,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1))
df$Flights.Per.Year<-  ifelse(df$Flights.Per.Year<= 6 ,'<6',
          ifelse(df$Flights.Per.Year<=12,'6-12',
```

```
                ifelse(df$Flights.Per.Year<=25,'13-25',
                    ifelse(df$Flights.Per.Year<=38  ,'26-38','38+'))))
table(df$Flights.Per.Year)

quantile(df$Loyalty,c(0,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1))
df$Loyalty<-  ifelse(df$Loyalty<= -0.75 ,'<-0.75',
                    ifelse(df$Loyalty<=-0.45,'-0.7501 to -0.45',
                        ifelse(df$Loyalty<=0.0588,'-0.451 to 0.0588',
                            ifelse(df$Loyalty<=0.3767  ,'0.059 to 0.3768','0.3768+'))))
table(df$Loyalty)

quantile(df$Loyalty,c(0,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1))
df$Loyalty<-  ifelse(df$Loyalty<= -0.75 ,'<-0.75',
                ifelse(df$Loyalty<=-0.45,'-0.7501 to -0.45',
                    ifelse(df$Loyalty<=0.0588,'-0.451 to 0.0588',
                        ifelse(df$Loyalty<=0.3767  ,'0.059 to 0.3768','0.3768+'))))
table(df$Loyalty)
quantile(df$Total.Freq.Flyer.Accts,c(0,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1))
df$Total.Freq.Flyer.Accts<-  ifelse(df$Total.Freq.Flyer.Accts==0 ,'0',
                ifelse(df$Total.Freq.Flyer.Accts==1,'1','1+'))

table(df$Total.Freq.Flyer.Accts)

quantile(df$Departure.Delay.in.Minutes,c(0,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1),na.rm
=TRUE)
df$Departure.Delay.in.Minutes<-  ifelse(df$Departure.Delay.in.Minutes<=5 ,'LESS THAN 5 MINS',
                            ifelse(df$Departure.Delay.in.Minutes<=10 ,'5 MINS TO 10 MINS',
                                ifelse(df$Departure.Delay.in.Minutes<=20 ,'10 MINS TO 20 MINS',
                                    ifelse(df$Departure.Delay.in.Minutes<=60 ,'20 MINS TO 60 MINS',
                                        ifelse(df$Departure.Delay.in.Minutess<=180,'60 MINS TO 180 MINS','180+
Minutes')))))

table(df$Departure.Delay.in.Minutes)
df$Departure.Delay.in.Minutes[which(is.na(df$Departure.Delay.in.Minutes)==TRUE)]<-'0-60 Mins'

quantile(df$Arrival.Delay.in.Minutes,c(0,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1),na.rm=T
RUE)
df$Arrival.Delay.in.Minutes<-  ifelse(df$Arrival.Delay.in.Minutes<=5 ,'LESS THAN 5 MINS',
                            ifelse(df$Arrival.Delay.in.Minutes<=10 ,'5 MINS TO 10 MINS',
                                ifelse(df$Arrival.Delay.in.Minutes<=20 ,'10 MINS TO 20 MINS',
                                ifelse(df$Arrival.Delay.in.Minutes<=60 ,'20 MINS TO 60 MINS',
                                ifelse(df$Arrival.Delay.in.Minutes<=180,'60 MINS TO 180 MINS','180+ Minutes')))))

table(df$Arrival.Delay.in.Minutes)
df$Arrival.Delay.in.Minutes[which(is.na(df$Arrival.Delay.in.Minutes)==TRUE)]<-'0-60 Mins'

quantile(df$Flight.time.in.minutes,c(0,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1),na.rm=TR
UE)
```

```
df$Flight.time.in.minutes<-  ifelse(df$Flight.time.in.minutes<=400 ,'LESS THAN 400 MINS',
                             ifelse(df$Flight.time.in.minutes<=1000 ,'400 MINS TO 1000 MINS','1000+ MINS'))
                            # ifelse(df$Flight.time.in.minutes<=20 ,'10 MINS TO 20 MINS',
                            # ifelse(df$Flight.time.in.minutes<=60 ,'20 MINS TO 60 MINS',
#)))

table(df$Flight.time.in.minutes)
df$Flight.time.in.minutes[which(is.na(df$Flight.time.in.minutes)==TRUE)]<-'60 MINS TO 180 MINS'

quantile(df$Flight.Distance,c(0,0.1,0.2,0.25,0.30,0.40,0.50,0.55,0.60,0.70,0.75,0.80,0.850,0.90,1),na.rm=TRUE)
df$Flight.Distance<-  ifelse(df$Flight.Distance<=400  ,'LESS THAN 400 MILES',
                             ifelse(df$Flight.Distance<=1000.0,'400 MILES TO 1000.0 MILES','1000.0+ MILES'))

table(df$Flight.Distance)
df$Flight.Distance[which(is.na(df$Flight.Distance)==TRUE)]<-'400 MILES TO 1000.0 MILES'

df_new<-df[,c("Partner.Name","Age","Gender","Airline.Status","Price.Sensitivity",

"Loyalty","Type.of.Travel","Total.Freq.Flyer.Accts","Class","Flights.Per.Year","Departure.Delay.in.Minutes","Arriv
al.Delay.in.Minutes"
          ,"Flight.time.in.minutes","Flight.Distance","Flight.cancelled","Likelihood.to.recommend")]

df_new$Departure.Delay.in.Minutes[which(df_new$Flight.cancelled=='Yes')] <- '0'
df_new$Arrival.Delay.in.Minutes[which(df_new$Flight.cancelled=='Yes')] <- '0'

#writing the tranformed data into csv
write.csv(df_new,"E:\\Disk partition\\Syracuse ADS\\Syracuse ADS\\1st Semester\\IST 687 Introduction to Data
Science\\project\\airline_customer_churner\\data.csv")

##### LOGISTIC REGRESSION #####
df_new<-read.csv("E:\\Disk partition\\Syracuse ADS\\Syracuse ADS\\1st Semester\\IST 687 Introduction to Data
Science\\project\\airline_customer_churner\\data_new2.csv")
df_new<-df_new[,-1]
library(caTools)

set.seed(88)
#splitting data into training and test data set for logistic regression
split <- sample.split(df_new$Likelihood.to.recommend, SplitRatio = 0.75)
train <- subset(df_new, split == TRUE)
test <- subset(df_new, split == FALSE)
#fitting training data into glm model
model <- glm(Likelihood.to.recommend ~.,family=binomial(link='logit'),data=train)
#output of logistic regression with respective beta coefficients
summary(model)
#fetching dominant features from the model with respect to p-values
anova(model, test="Chisq")

#Calculating model accuracy
```

```
fitted.results <- predict(model,newdata=subset(test),type='response')
fitted.results <- ifelse(fitted.results > 0.5,1,0)

misClasificError <- mean(fitted.results != test$Likelihood.to.recommend,na.rm=TRUE)
print(paste('Accuracy',1-misClasificError))

###"Accuracy 0.756420233463035"

############################threshold variation##############################
#model optimization by threshold variation
fitted1.results_lr <- predict(model,newdata=subset(test),type='response')
fitted1.results_lr <- ifelse(fitted1.results_lr > 0.5,1,0)

misClasificError1 <- mean(fitted1.results_lr != test$Likelihood.to.recommend,na.rm=TRUE)
print(paste('Accuracy',1-misClasificError1))
#"Accuracy 0.75875486381323"

y_train <- dresstrain$Likelihood.to.recommend

x_train <-dresstrain%>%
select(Partner.Name,Age,Gender,Airline.Status,Price.Sensitivity,Loyalty,Type.of.Travel,Total.Freq.Flyer.Accts

,Class,Flights.Per.Year,Departure.Delay.in.Minutes,Arrival.Delay.in.Minutes,Flight.time.in.minutes
                    ,Flight.Distance,Flight.cancelled) %>% data.matrix()

#calculating generalization performance measure by AUC
library(ROCR)
p <- predict(model, newx=subset(x_train), type="response")
pr <- prediction(p, y_train)
auc <- performance(pr, measure = "auc")
auc
#AUC Score
#0.8428508
############################hyper parameter tuning#####################################
library('caret')
set.seed(88)

split <- sample.split(df_new$Likelihood.to.recommend, SplitRatio = 0.75)
dresstrain <- subset(df_new, split == TRUE)
dresstest <- subset(df_new, split == FALSE)
modelLookup('glm')
method = 'glm'
library(glmnet)
library(tidyverse)

y <- dresstrain$Likelihood.to.recommend
```

```
x<-dresstrain%>%
select(Partner.Name,Age,Gender,Airline.Status,Price.Sensitivity,Loyalty,Type.of.Travel,Total.Freq.Flyer.Accts
                  ,Class,Flights.Per.Year,Departure.Delay.in.Minutes,Arrival.Delay.in.Minutes,Flight.time.in.minutes
                  ,Flight.Distance,Flight.cancelled) %>% data.matrix()
typeof(x)
str(x)

# set up the grid to find the minimum lambda
tuneGrid <- expand.grid(.alpha = seq(0, 1, 0.05), .lambda = seq(0, 2, 0.05))
## 10-fold CV ##
#works with a single factor variable  (ignore warnings based on small sample size)
mod_elnet<-cv.glmnet(x,y, data=dresstrain, method="glmnet",
                 family="binomial", tuneGrid = tuneGrid, metric = "ROC",type.measure = "auc")
mod_elnet
#min lmabda
#0.0007238316
cbind(mod_elnet$lambda,mod_elnet$cvm)

#best parameter
mod_elnet$lambda.min

#best coefficient
coef(mod_elnet, s = "lambda.min")

#cross validation
#x_test<-dresstest %>%
select(Partner.Name,Age,Gender,Airline.Status,Price.Sensitivity,Loyalty,Type.of.Travel,Total.Freq.Flyer.Accts
#                   ,Class,Flights.Per.Year,Departure.Delay.in.Minutes,Arrival.Delay.in.Minutes,Flight.time.in.minutes
#                   ,Flight.Distance,Flight.cancelled) %>% data.matrix()
pred = predict(mod_elnet, newx = x_test, type = 'response',s ="lambda.min")

#install.packages('AUC') #uncomment and run to install packe for AUC calculation
library(AUC)
library(ROC)
help(AUC)
auc(y,response)
y_test <- dresstest$Likelihood.to.recommend

x_test <-dresstest%>%
select(Partner.Name,Age,Gender,Airline.Status,Price.Sensitivity,Loyalty,Type.of.Travel,Total.Freq.Flyer.Accts
                  ,Class,Flights.Per.Year,Departure.Delay.in.Minutes,Arrival.Delay.in.Minutes,Flight.time.in.minutes
                  ,Flight.Distance,Flight.cancelled) %>% data.matrix()
fitted.results <- predict(mod_elnet,newx=subset(x_test),type='response')
fitted.results <- ifelse(fitted.results > 0.5,1,0)

misClasificError <- mean(fitted.results != y_test,na.rm=TRUE)
print(paste('Accuracy',1-misClasificError))
"Accuracy 0.702334630350194"
```

```
library(ROCR)
p <- predict(mod_elnet, newx=subset(x_test), type="response")
pr <- prediction(p, y_test)
auc <- performance(pr, measure = "auc")
auc
# 0.7657839
```
########################## best model logistic determined ################################

#Random forest

```
df_new<-read.csv("E:\\Disk partition\\Syracuse ADS\\Syracuse ADS\\1st Semester\\IST 687 Introduction to Data Science\\project\\airline_customer_churner\\data_new2.csv")
df_new<-df_new[,-1]
# Create model with default paramters
library(randomForest)
library(mlbench)
library(caret)

par(mar = c(5, 8, 1, 1))
library(randomForest)

set.seed(88)
#splitting data into training and test data set for logistic regression
split <- sample.split(df_new$Likelihood.to.recommend, SplitRatio = 0.75)
train <- subset(df_new, split == TRUE)
test <- subset(df_new, split == FALSE)

#function that returns random forest feature importance plots
create_rfplot <- function(rf_default, type){

 imp <- importance(rf_default, type = type, scale = F)

 featureImportance <- data.frame(Feature = row.names(imp), Importance = imp[,1])

 p <- ggplot(featureImportance, aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "#53cfff", width = 0.65) +
  coord_flip() +
  theme_light(base_size = 20) +
  theme(axis.title.x = element_text(size = 15, color = "black"),
      axis.title.y = element_blank(),
      axis.text.x  = element_text(size = 15, color = "black"),
      axis.text.y  = element_text(size = 15, color = "black"))
 return(p)
}

#####Random forest model after parameter tuning the best model performs with 200 trees
rf1 <- randomForest(
 Likelihood.to.recommend ~ .,
 ntree = 200,
```

```
  data = train,
  nodesize = 1,
  replace = FALSE,
  importance = TRUE
)
#printing output of random forest
print(rf1)
#plotting feature importance graph of random forest
create_rfplot(rf1, type = 2)

#Calculating AUC for the model
library(ROCR)
p <- predict(rf1, newdata=subset(test), type="response")
pr <- prediction(p, test$Likelihood.to.recommend)
auc <- performance(pr, measure = "auc")
print(auc)
#AUC 0.8471364

#calculating Accuracy of the model
fitted.results_rf <- predict(rf1,newdata=subset(test),type='response')
fitted.results_rf <- ifelse(fitted.results_rf > 0.4,1,0)

misClasificError <- mean(fitted.results_rf != test$Likelihood.to.recommend,na.rm=TRUE)
print(paste('Accuracy',1-misClasificError))
#"Accuracy 0.761089494163424"

# XGBOOST
df_new<-read.csv("E:\\Disk partition\\Syracuse ADS\\Syracuse ADS\\1st Semester\\IST 687 Introduction to Data
Science\\project\\airline_customer_churner\\data_new2.csv")
df_new<-df_new[,-1]
View(df_new)
require(xgboost)
require(Matrix)
require(data.table)
if (!require('vcd')) install.packages('vcd')
library(xgboost)

library(caTools)

set.seed(88)
#splitting the data into test and train data set
split <- sample.split(df_new$Likelihood.to.recommend, SplitRatio = 0.75)
train <- subset(df_new, split == TRUE)
test <- subset(df_new, split == FALSE)
set.seed(123)
# for reproducibility
set.seed(123)
```

```
#feature engineering for xgboost converting features into sparse matrix
sparse_matrix <- sparse.model.matrix(Likelihood.to.recommend~.-1, data = train)
output_vector = train$Likelihood.to.recommend

#xgboost model
bst <- xgboost(data = sparse_matrix, label = output_vector, max.depth = 4,
            eta = 1, nthread = 2, nrounds = 10,objective = "binary:logistic")

#determining and plotting top 10 feature importances
importance <- xgb.importance(feature_names = sparse_matrix@Dimnames[[2]], model = bst)
importanceRaw <- xgb.importance(feature_names = sparse_matrix@Dimnames[[2]], model = bst, data =
sparse_matrix, label = output_vector)
xgb.plot.importance(importance_matrix = importanceRaw[1:10])


y_train <- train$Likelihood.to.recommend

x_train <-train%>%
select(Partner.Name,Age,Gender,Airline.Status,Price.Sensitivity,Loyalty,Type.of.Travel,Total.Freq.Flyer.Accts

,Class,Flights.Per.Year,Departure.Delay.in.Minutes,Arrival.Delay.in.Minutes,Flight.time.in.minutes
                        ,Flight.Distance,Flight.cancelled) %>% data.matrix()


dtrain <- xgb.DMatrix(x_train, label = y_train)
y_test <- test$Likelihood.to.recommend

x_test <-test%>%
select(Partner.Name,Age,Gender,Airline.Status,Price.Sensitivity,Loyalty,Type.of.Travel,Total.Freq.Flyer.Accts
                    ,Class,Flights.Per.Year,Departure.Delay.in.Minutes,Arrival.Delay.in.Minutes,Flight.time.in.minutes
                    ,Flight.Distance,Flight.cancelled) %>% data.matrix()

#Calculating generalization performance measue AUC
sparse_matrix_test <- sparse.model.matrix(Likelihood.to.recommend~.-1, data = test)
output_vector_test = test$Likelihood.to.recommend
pred <- predict(bst, sparse_matrix_test)
pr <- prediction(pred, output_vector_test)
auc <- performance(pr, measure = "auc")
print(auc)

#Calculating model Accuracy
fitted.results_gbm <- predict(bst,sparse_matrix_test)
fitted.results_gbm <- ifelse(fitted.results_gbm > 0.5,1,0)

misClasificError <- mean(fitted.results_gbm != test$Likelihood.to.recommend,na.rm=TRUE)
print(paste('Accuracy',1-misClasificError))
#"Accuracy 0.754474708171206
# ############################boost 1
```

```
#
# bst_new <- xgboost(data = dtrain, max.depth = 2, eta = 1, nthread = 2, nrounds = 2, objective = "binary:logistic",
verbose = 2)
#
#
# y_test <- test$Likelihood.to.recommend
#
# x_test <-test%>%
select(Partner.Name,Age,Gender,Airline.Status,Price.Sensitivity,Loyalty,Type.of.Travel,Total.Freq.Flyer.Accts
#
,Class,Flights.Per.Year,Departure.Delay.in.Minutes,Arrival.Delay.in.Minutes,Flight.time.in.minutes
#                       ,Flight.Distance,Flight.cancelled) %>% data.matrix()
# pred <- predict(bst_new, x_test)
# pr <- prediction(pred, y_test)
# auc <- performance(pr, measure = "auc")
# print(auc)
#
#
# ###################################3boost 2
bst2 <- xgboost(data = dtrain, label = output_vector, max.depth = 4,
          eta = 1, nthread = 2, nrounds = 10,objective = "binary:logistic")
y_test <- test$Likelihood.to.recommend

x_test <-test%>%
select(Partner.Name,Age,Gender,Airline.Status,Price.Sensitivity,Loyalty,Type.of.Travel,Total.Freq.Flyer.Accts
                    ,Class,Flights.Per.Year,Departure.Delay.in.Minutes,Arrival.Delay.in.Minutes,Flight.time.in.minutes
                    ,Flight.Distance,Flight.cancelled) %>% data.matrix()
pred <- predict(bst2, x_test)
pr <- prediction(pred, y_test)
auc <- performance(pr, measure = "auc")
print(auc)

# dtest <- xgb.DMatrix(data = x_test, label=y_test)
# watchlist <- list(train=dtrain, test=dtest)
#
# bst3 <- xgb.train(data=dtrain, max.depth=2, eta=1, nthread = 2, nrounds=2, watchlist=watchlist, eval.metric =
"auc", eval.metric = "logloss", objective = "binary:logistic")
# bst4 <- xgb.train(data=dtrain, max.depth=2, eta=1, nthread = 2, nrounds=2, watchlist=watchlist, objective =
"binary:logistic")
# bst5 <- xgb.train(data=dtrain, max.depth=2, eta=1, nthread = 2, nrounds=2, watchlist=watchlist, eval.metric =
"auc", eval.metric = "logloss", objective = "binary:logistic")
# head(importance)
```

```
#installing packages for clustering
install.packages("clustMixType")
df_new<-read.csv("E:\\Disk partition\\Syracuse ADS\\Syracuse ADS\\1st Semester\\IST 687 Introduction to Data Science\\project\\airline_customer_churner\\data_new2.csv")
df_new<-df_new[,-1]
library(clustMixType)
install.packages("wss")
library(wss)
data <- df_new
# Elbow Method for finding the optimal number of clusters
set.seed(123)
# Compute and plot wss for k = 2 to k = 15.
k.max <- 15
data <- na.omit(data) # to remove the rows with NA's

#K prototype clustering
wss <- sapply(1:k.max,
              function(k){kproto(data, k)$tot.withinss})
wss
plot(1:k.max, wss,
    type="b", pch = 19, frame = FALSE,
    xlab="Number of clusters K",
    ylab="Total within-clusters sum of squares")

t1<-Sys.time()
#elbow curve hits descent at cluster 4 hence formulating relations with 4 clusters
no_of_clusters<-4
set.seed(12345)
fit_clust2 <- kproto(data, no_of_clusters, lambda = 1.804845  ,iter.max=60)
print("Time taken for k-prototypes")
t2<-Sys.time()
t2-t1

#viewing cluster plots
a<-clprofiles(fit_clust2, data)

# binning data Apriori
rm(list=ls())
df_nps <- jsonlite::fromJSON("./fall2019-survey-M09.json")

#Read the data output from bucketing and cleaning R file
df_new<-read.csv("data_new2.csv")
df_new<-df_new[,-1]
glimpse(df_new)
dim(df_new)
df_new <- data.frame(apply(df_new,2,as.factor))

# use the top 10 important features in apriori
```

```
df_new1 = df_new[,c(1,4,5,7,8,9,10,11,12,13,14,15,16)]
glimpse(df_new1)

# rename the columns
gsub("\\.","_",names(df_new1))
names(df_new1) <- gsub("\\.","_",names(df_new1))

apply(apply(df_new1, 2, is.na), 2, which)
df_new1$type_passenger[which(is.na(df_new1$type_passenger))] = "passive"

# create transantions data type
df_newX <- as(df_new1, "transactions")
unique(df_new1$Likelihood_to_recommend)


inspect(df_newX)
itemFrequency(df_newX)
itemFrequencyPlot(df_newX)

# Apply apriori algorithm
ruleset <- apriori(df_newX,
                   # Specify threshold of 0.005 for support, and 0.5 for confidence. That is, show only those values that are
higher than the thresholds.
                   parameter = list(support=0.05, confidence = 0.05, minlen = 3),
                   # Specify the rhs as "Survived = Yes", and all the rest of the variables of the transaction as lhs of the
equation
                   appearance = list(default="lhs", rhs = ("Likelihood_to_recommend= 0"))) # 0 for detractors, and 1 for
promoters

#inspect the rules from apriori
inspect(ruleset)
inspectDT(ruleset)


# Type of travel: Personal travel
ggplot(df_nps_complete, aes(Airline.Status))+
  geom_histogram(aes(fill = type_passenger), bins = 10, binwidth = 0.5, color = 'Black', stat = 'count') +
  theme(axis.text.x = element_text(angle = 0, hjust = 1))


# type_passenger in df_new
dim(df_new)
dim(df_nps)
df_new$type_passenger <- cut(df_nps$Likelihood.to.recommend,
                             breaks = c(-1,7,9,Inf),
                             labels = c("detractor","passive","promoter"))

table(df_new$type_passenger)
```

```
df_new$type_passenger[which(is.na(df_new$type_passenger))] = "passive"
df_new$type_passenger <- factor(df_new$type_passenger, ordered = TRUE)

names(df_new) <- gsub("\\.","_",names(df_new))

# Create recommender type
df_nps$type_passenger <- cut(df_nps$Likelihood.to.recommend,
                        breaks = c(-1,7,9,Inf),
                        labels = c("detractor","passive","promoter"))

table(df_nps$type_passenger)
df_nps$type_passenger[which(is.na(df_nps$type_passenger))] = "passive"
df_nps$type_passenger <- factor(df_nps$type_passenger, ordered = TRUE)


df_nps1 <- copy(df_nps)
names(df_nps1) <- gsub("\\.","_",names(df_nps1))

df_nps1$promoter <- ifelse(df_nps1$type_passenger == 'promoter',1,0)
df_nps1$detractor <- ifelse(df_nps1$type_passenger == 'detractor',1,0)
df_nps1$passive <- ifelse(df_nps1$type_passenger == 'passive',1,0)
df_nps1$not_promoter <- df_nps1$passive+df_nps1$detractor

glimpse(df_nps1)


library(sqldf)

# Create data of all unique flights based in origin city and destination city
df_nps2 <- sqldf("
            SELECT d1.Origin_City, d1.Destination_City, d1.Origin_State, d1.Destination_State, d1.olong, d1.olat,
d1.dlong, d1.dlat, COUNT(olong) As Count, SUM(promoter) as promoter, SUM(not_promoter) as not_promoter,
SUM(detractor) as detractor, SUM(passive) as passive, SUM(promoter)/COUNT(olong) as promoter_per,
SUM(not_promoter)/COUNT(olong) as not_promoter_per
            FROM df_nps1 AS d1
            GROUP BY  d1.Origin_City, d1.Destination_City, d1.Origin_State, d1.Destination_State, d1.olong,
d1.olat, d1.dlong, d1.dlat")

View(df_nps2)

# Create the dataset for cities with the most number of flights
df_nps2_busy_origin <- sqldf("
                SELECT d1.Origin_City, COUNT(d1.Destination_City) as busy
                FROM df_nps2 as d1
                GROUP BY d1.Origin_City
                ")
```

```
View(df_nps2_busy_origin)


df_busy <- sqldf("
            SELECT d1.*
            FROM df_nps1 AS d1 INNER JOIN df_nps2_busy_origin d2 ON d1.Origin_City = d2.Origin_City
            WHERE d2.busy > 70
            ")

View(df_busy)

#Plot the histogram of histogram of delay
ggplot(df_busy, aes(Departure_Delay_in_Minutes))+
  geom_histogram(aes(fill = type_passenger),bins = 10, color ='black')

ggplot(df_nps, aes(type_passenger))+
  geom_histogram(stat = 'count')

# Create dataset for different good flights
df_good <- sqldf("
            SELECT *
            FROM df_nps1 d1
            WHERE d1.Origin_City = 'Buffalo, NY' and d1.Destination_City = 'New York, NY'")

View(df_good)

df_good2 <- sqldf("
            SELECT *
            FROM df_nps1 d1
            WHERE d1.Origin_City = 'San Diego, CA' and d1.Destination_City = 'Los Angeles, CA'")

View(df_good2)

df_good3 <- sqldf("
            SELECT d1.*
            FROM df_nps1 d1 LEFT JOIN df_nps2 d2 ON d1.Origin_City = d2.Origin_City and d1.Destination_City
= d2.Destination_City
            WHERE d2.promoter_per > 0.27 and d2.Count > 10")

View(df_good3)

# Create datasets for bad flights
df_bad <- sqldf("
            SELECT *
            FROM df_nps1 d1
            WHERE d1.Origin_City = 'Houston, TX'")

View(df_bad)
```

```
df_bad2 <- sqldf("
            SELECT d1.*
            FROM df_nps1 d1 LEFT JOIN df_nps2 d2 ON d1.Origin_City = d2.Origin_City and d1.Destination_City
= d2.Destination_City
            WHERE d2.not_promoter_per = 1 and d2.Count > 10")

View(df_bad2)

#plots for comparison

dim(df_bad2)
dim(df_good3)
ggplot(df_bad2, aes(Age))+
  geom_histogram(aes(fill = Airline_Status), bins = 10, color = 'Black') +
  ggtitle("Bad Flights")+
  theme(axis.text.x = element_text(angle = 0))

ggplot(df_good3, aes(Age))+
  geom_histogram(aes(fill = Airline_Status), bins = 10, color = 'Black') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 0))

# class
ggplot(df_bad2, aes(Class))+
  geom_histogram(aes(fill = Airline_Status), bins = 10, color = 'Black', stat = 'count') +
  ggtitle("Bad Flights")+
  theme(axis.text.x = element_text(angle = 0))

ggplot(df_good3, aes(Class))+
  geom_histogram(aes(fill = Airline_Status), bins = 10, color = 'Black', stat = 'count') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 0))

# Type of travel
ggplot(df_bad2, aes(Type_of_Travel))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black', stat = 'count') +
  ggtitle("Bad Flights")+
  theme(axis.text.x = element_text(angle = 0))+ylim(0, 400)

ggplot(df_good3, aes(Type_of_Travel))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black', stat = 'count') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 0))+ylim(0, 400)

# Flight time
ggplot(df_bad2, aes(Flight_time_in_minutes))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black') +
```

```
  ggtitle("Bad Flights")+
  theme(axis.text.x = element_text(angle = 0))+ylim(0,200)

ggplot(df_good3, aes(Flight_time_in_minutes))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 0))+ylim(0,200)

#flight distance
ggplot(df_bad2, aes(Flight_Distance))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black') +
  ggtitle("Bad Flights")+
  theme(axis.text.x = element_text(angle = 0))

ggplot(df_good3, aes(Flight_Distance))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 0))

#Partner.Name
ggplot(df_bad2, aes(reorder(Partner_Name,Partner_Name)))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black', stat = 'count') +
  ggtitle("Bad Flights")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+ylim(0,125)

ggplot(df_good3, aes(reorder(Partner_Name,Partner_Name)))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black', stat = 'count') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 90, hjust=1))+ylim(0,125)

ggplot(df_nps, aes(reorder(Partner.Name,Partner.Name)))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black', stat = 'count') +
  ggtitle("Total Flights")+
  theme(axis.text.x = element_text(angle = 90, hjust=1))


#FLights per year
ggplot(df_nps, aes(x = Flights.Per.Year, y = Loyalty))+
  geom_point(color = "blue")+geom_vline(xintercept = 28, linetype = 'dashed', color = 'red', size = 1.5)+
  geom_hline(yintercept = -0.5, linetype = 'dashed', color = 'red', size = 1.5)+
  ggtitle("Flights per year vs. Loyalty")


ggplot(df_nps[df_nps$Type.of.Travel == 'Mileage tickets',], aes(Flights.Per.Year))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 90, hjust=1))
```

```
ggplot(df_good3, aes(Flight_Distance))+
  geom_histogram(aes(fill = type_passenger), bins = 10, color = 'Black') +
  ggtitle("Good Flights")+
  theme(axis.text.x = element_text(angle = 0))
```