Classification

Meinhard Capucao, Khang Thai

This data-set contains information about customers and their satisfaction with an airline organization.

Here is a link to the dataset.

Logistic regression and the **Naive Bayes** model for classification. Each model has its own strengths and weaknesses described below.

Logistic Regression

Logistic regression is similar to Linear regression in that it has a “linear” relationship between the predictor value and the target value. The only difference is that the values used are qualitative instead of quantitative. A linear model for classification makes decision boundaries where observations are to be sorted later on. We use logistic regression in order to predict a qualitative outcome based on the given predictors. Logistic regression is able to separate classes if they are linearly separable, and gives a nice probabilistic output. However, a weakness is that it is prone to underfitting, and how it is assumed that there is a linear relationship between the x and y already.

Bayes Model

Naive Bayes model is a simple algorithm based on conditional probability placed on a probability table.The Naive Bayes model is based on the Bayes theorem which states that given the conditional probability of an event, what is the probability of another event will occur. Bayes strength lies in the fact that it is easily created while its simplicity is also its weakness because Bayes model will generate a fast prediction but will fall behind against other models that are trained with better data.

Install Required Packages

We want to install tidyverse in order to read in the csv file.

```
install.packages("tidyverse",repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/meinc/AppData/Local/R/win-library/4.2'
## (as 'lib' is unspecified)
```

```
## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\meinc\AppData\Local\Temp\RtmpUrn6qt\downloaded_packages
```

```
library(tidyverse)
```

```
## ── Attaching packages
## ───────────────────────────────────────
## tidyverse 1.3.2 ──
```

```
## ✔ ggplot2 3.3.6      ✔ purrr   0.3.4
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.2.1      ✔ stringr 1.4.1
## ✔ readr   2.1.2      ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
Invistico_Airline <- read_csv("Invistico_Airline.csv")
```

```
## Rows: 129880 Columns: 23
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (5): satisfaction, Gender, Customer Type, Type of Travel, Class
## dbl (18): Age, Flight Distance, Seat comfort, Departure/Arrival time conveni...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

In this project, we want to see the effects of average customer rating on customer satisfaction: whether they are “satisfied” or “dissatisfied”.

Divide into Train and Test Data

First, we divide our data into train and test. We set a unique seed to be able to replicate our data, then divide it into 80% training and 20% test.

```
set.seed(1234)
i <- sample(1:nrow(Invistico_Airline), nrow(Invistico_Airline)*0.8, replace = FALSE)
train <- Invistico_Airline[i,]
test <- Invistico_Airline[-i,]
```

Data Exploration

We can learn more about what the data set contains. The most basic way is to observe the dataset with the head() function, printing out the first 6 rows. Notice that our main target, satisfaction, accepts characters of “satisfied”, with the other value being “dissatisfied”. We want to sum all of the ratings and average it, to be more concise with this experiment.

```
head(Invistico_Airline)
```

satisfaction	Gender	Customer Type	…	Type of Travel	Class	Flight Distance	Seat comfort
<chr>	<chr>	<chr>	<dbl>	<chr>	<chr>	<dbl>	<dbl>
satisfied	Female	Loyal Customer	65	Personal Travel	Eco	265	0
satisfied	Male	Loyal Customer	47	Personal Travel	Business	2464	0
satisfied	Female	Loyal Customer	15	Personal Travel	Eco	2138	0
satisfied	Female	Loyal Customer	60	Personal Travel	Eco	623	0
satisfied	Female	Loyal Customer	70	Personal Travel	Eco	354	0
satisfied	Male	Loyal Customer	30	Personal Travel	Eco	1894	0

6 rows | 1-8 of 23 columns

There are 14 columns for different ratings, ranging from seat comfort to food and drink. The minimum rating is 0, and the maximum is 5. We will **create two new columns** named **ratingSum** and **ratingMean** for both the train and test data. The mean value is the ratingSum divided by total number of rating factors.

```
train$ratingSum <- as.numeric(apply(train[,8:21], 1, sum))
test$ratingSum <- as.numeric(apply(test[,8:21], 1, sum))

test$ratingMean <- c(test$ratingSum/14)
train$ratingMean <- c(train$ratingSum/14)

head(train)
```

satisfaction	Gender	Customer Type	…	Type of Travel	Class	Flight Distance	Seat comfort
<chr>	<chr>	<chr>	<dbl>	<chr>	<chr>	<dbl>	<dbl>
satisfied	Female	Loyal Customer	32	Business travel	Business	4906	1
satisfied	Female	Loyal Customer	43	Business travel	Business	334	1
dissatisfied	Male	disloyal Customer	66	Business travel	Business	1724	1
satisfied	Female	Loyal Customer	69	Personal Travel	Eco	433	4
dissatisfied	Female	Loyal Customer	10	Personal Travel	Eco Plus	1566	4
satisfied	Female	Loyal Customer	36	Business travel	Eco	1227	4

6 rows | 1-8 of 25 columns

Next, we sort by rating so we can see the **lowest average** rating mean from all customers. We can get insight into the overall data, if the average rating mean for a customer is as low as 1.1, but they are satisfied, there must be other factors that influence customers satisfication within an airline.

Here, we also cut down the columns to only those deemed necessary, **Satisfication**, **Gender**, **Customer Type**, and **ratingMean**. We sort it again to show the first 200 entries from ‘sortByRating2’. Notice all from customers with the lowest rating mean, yet some of them are satisfied.

We make sure to do this for the test data as well so unnecessary columns are cut from both training and test data.

```
sortByRating <- train[order(train$ratingMean),]
head(sortByRating)
```

satisfaction	Gender	Customer Type	…	Type of Travel	Class	Flight Distance	Seat comfort
<chr>	<chr>	<chr>	<dbl>	<chr>	<chr>	<dbl>	<dbl>
satisfied	Female	Loyal Customer	51	Personal Travel	Eco	1351	0
satisfied	Male	Loyal Customer	32	Business travel	Eco	2077	0
dissatisfied	Female	Loyal Customer	57	Business travel	Business	1942	1
satisfied	Female	Loyal Customer	24	Personal Travel	Eco	1808	0
dissatisfied	Female	Loyal Customer	28	Business travel	Eco Plus	1491	1
dissatisfied	Female	Loyal Customer	53	Business travel	Business	3510	1

6 rows | 1-8 of 25 columns

```
sortByRating2 <- train[, c('satisfaction', 'Gender', 'Customer Type', 'ratingMean')]
sortByRating2 <- sortByRating2[order(sortByRating2$ratingMean),]
head(sortByRating2[,1:4],200)
```

satisfaction	Gender	Customer Type	ratingMean
<chr>	<chr>	<chr>	<dbl>
satisfied	Female	Loyal Customer	1.071429
satisfied	Male	Loyal Customer	1.071429
dissatisfied	Female	Loyal Customer	1.142857
satisfied	Female	Loyal Customer	1.142857
dissatisfied	Female	Loyal Customer	1.142857
dissatisfied	Female	Loyal Customer	1.142857
satisfied	Female	Loyal Customer	1.142857
dissatisfied	Male	Loyal Customer	1.214286
dissatisfied	Male	disloyal Customer	1.214286
dissatisfied	Female	Loyal Customer	1.214286

1-10 of 200 rows Previous 1 2 3 4 5 6 ... 20 Next

```
sortByRating2Test <- test[, c('satisfaction', 'Gender', 'Customer Type', 'ratingMean')]
sortByRating2Test <- sortByRating2Test[order(sortByRating2Test$ratingMean),]
```

Here, we wanted to see how many customers are loyal, and how many are disloyal. Notice how there are a lot more loyal customers than disloyal. Male

```
length(which(sortByRating2$`Customer Type` == "Loyal Customer"))
```

```
## [1] 84965
```

```
length(which(sortByRating2$`Customer Type` == "disloyal Customer"))
```

```
## [1] 18939
```

```
length(which(sortByRating2$`Gender` == "Male"))
```

```
## [1] 51163
```

```
length(which(sortByRating2$`Gender` == "Female"))
```

```
## [1] 52741
```

Lastly, we wanted to have a basic observation on why customers with really low ratings still say they are satisfied. We thought that being a disloyal customer could've been a big part, so we filtered the table to only show disloyal customers. We saw that for the most part, disloyal customers were dissatisfied with overall service. There were over 3x (14,577) customers who were disloyal and dissatisfied than customers who were loyal and satisfied (4552).

```
sortByRating3 = subset(sortByRating2, sortByRating2$`Customer Type` == "disloyal Customer")
head(sortByRating3[,1:4],200)
```

satisfaction	Gender	Customer Type	ratingMean
<chr>	<chr>	<chr>	<dbl>
dissatisfied	Male	disloyal Customer	1.214286
dissatisfied	Male	disloyal Customer	1.214286
dissatisfied	Female	disloyal Customer	1.285714
dissatisfied	Female	disloyal Customer	1.285714
satisfied	Female	disloyal Customer	1.285714
dissatisfied	Male	disloyal Customer	1.285714
satisfied	Male	disloyal Customer	1.285714
dissatisfied	Male	disloyal Customer	1.357143
dissatisfied	Female	disloyal Customer	1.357143
dissatisfied	Female	disloyal Customer	1.357143

1-10 of 200 rows Previous 1 2 3 4 5 6 ... 20 Next

```
count(subset(sortByRating3, sortByRating3$satisfaction == "dissatisfied"))
```

n
<int>
14387

1 row

```
count(subset(sortByRating3, sortByRating3$satisfaction == "satisfied"))
```

n
<int>
4552

1 row

Factors

Here we make satisfcation, gender, and customer type into factors.

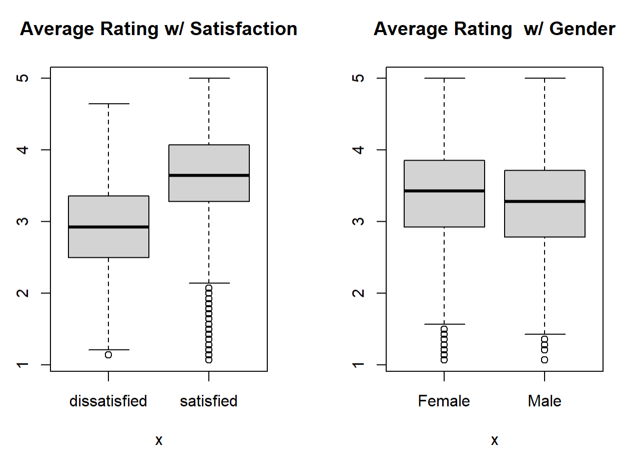
```
sortByRating2$satisfaction <- factor(sortByRating2$satisfaction)
sortByRating2$Gender <- factor(sortByRating2$Gender)
sortByRating2$`Customer Type` <- factor(sortByRating2$`Customer Type`)
```

Plotting the data

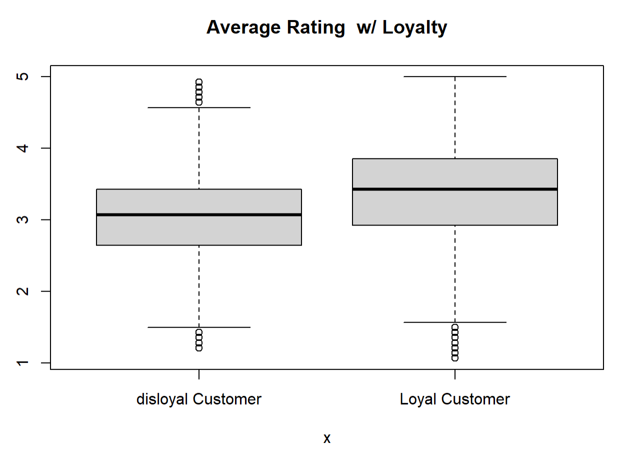
The first set of graphs were a box plot. We can conclude that customers who are satisfied tend to have higher average mean ratings of all categories. However, for satisfied customers, the graph considered that anyone with a mean rating of 2.0 or below to be an outlier. For gender, male and female were pretty similar, however females tend to have a slightly greater average rating. Loyal customers tend to have a higher rating, but not as much as we expected. Values below a 1.5 were considered outliers for most the graphs.

We can see that through the summary() function, the average mean rating was 3.310 for all customers.

```
par(mfrow=c(1,2))
plot(sortByRating2$satisfaction, sortByRating2$ratingMean, main = "Average Rating w/ Satisfaction", ylab = "")
plot(sortByRating2$Gender, sortByRating2$ratingMean, main = "Average Rating  w/ Gender", ylab = "")
```




```
par(mfrow=c(1,1))
plot(sortByRating2$`Customer Type`, sortByRating2$ratingMean, main = "Average Rating  w/ Loyalty", ylab = "")
```



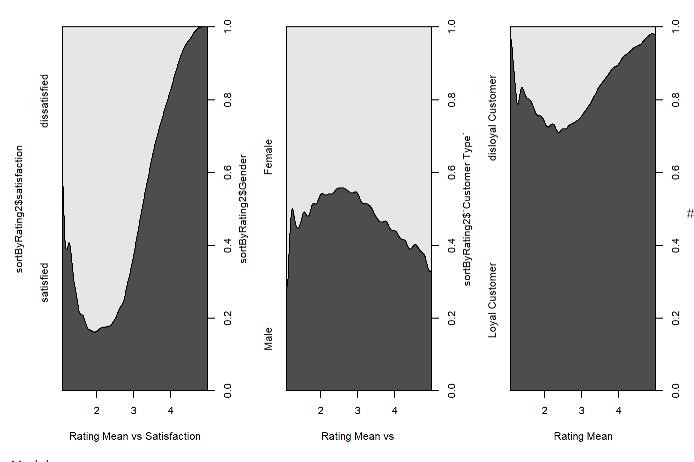

```
summary(sortByRating2)
```

```
##        satisfaction      Gender                Customer Type     ratingMean
##  dissatisfied:47000   Female:52741   disloyal Customer:18939   Min.   :1.071
##  satisfied   :56904   Male  :51163   Loyal Customer   :84965   1st Qu.:2.857
##                                                                Median :3.357
##                                                                Mean   :3.310
##                                                                3rd Qu.:3.786
##                                                                Max.   :5.000
```

The second set of graphs is a conditional density plot for all three factors. There are similar observations as the box plot.

```
par(mfrow=c(1,3))

cdplot(sortByRating2$satisfaction~sortByRating2$ratingMean, xlab = "Rating Mean vs Satisfaction")
cdplot(sortByRating2$Gender~sortByRating2$ratingMean, xlab = "Rating Mean vs ")
cdplot(sortByRating2$`Customer Type`~sortByRating2$ratingMean, xlab = "Rating Mean")
```



Building our Logistic Regression

Model

This model only uses ratingMean as a predictor for satisfcation. We use the glm() function for logistic regression with the parameter family=binomial.

```
glm1 <- glm(satisfaction~ratingMean, data=sortByRating2, family=binomial)
summary(glm1)
```

```
##
## Call:
## glm(formula = satisfaction ~ ratingMean, family = binomial, data = sortByRating2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4128  -0.9182   0.4382   0.8780   2.8851
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.24664    0.04504  -138.7   <2e-16 ***
## ratingMean   1.96032    0.01359   144.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 143096  on 103903  degrees of freedom
## Residual deviance: 112951  on 103902  degrees of freedom
## AIC: 112955
##
## Number of Fisher Scoring iterations: 4
```

Logistical Regression Model Summary

Based on the summary table, the information it shows is that with every one-unit change in ratingMean, the likelihood of satisfaction is increased by 1.96 with a standard error of .01 for each prediction. The difference between the Null deviance and the Residual deviance tells us that the model is a good fit. Null deviance tells us the value when we have no variables while the Residual deviance tells us the value when we have all variables taken into account. The bigger the difference the better the model.

Here, we build a table that states what our model predicted and the actual outcome. The prediction column on the left is our model's prediction, and the row going to the right is the actual values. We can see that our model predicted 5040 correct outcomes for when the customer is dissatisfied, and 3989 for when it is satisfied. Our model struggled when it predicted it was satisfied, but the customer was in reality dissatisfied. There could be many reasons, such as more predictors or a better model needed.

```
p1 <- predict(glm1, newdata=sortByRating2Test, type = "response")
prediction <- ifelse(p1>0.5, "dissatisfied", "satisfied")
table(prediction, test$satisfaction)
```

```
##
## prediction     dissatisfied satisfied
##   dissatisfied         5040     10194
##   satisfied            6753      3989
```

Our model seems to have relatively low accuracy, so let's check the number value.

```
mean(prediction==test$satisfaction)
```

```
## [1] 0.3475901
```

34% accuracy. This could be better, but for a logistical regression function with one predictor, it's not the worst. Let's move on to creating a Naive Bayes Model.

Building our Naive Bayes Model

Make sure to do install.packages(“e1071”), then load the package. We create the naive bayes model using the package we just downloaded.

```
library(e1071)
nb1 <- naiveBayes(satisfaction~., data=sortByRating2)
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
## dissatisfied    satisfied
##    0.4523406    0.5476594
##
## Conditional probabilities:
##               Gender
## Y                 Female      Male
##   dissatisfied 0.3917234 0.6082766
##   satisfied    0.6032968 0.3967032
##
##               Customer Type
## Y               disloyal Customer Loyal Customer
##   dissatisfied        0.30610638     0.69389362
##   satisfied           0.07999438     0.92000562
##
##               ratingMean
## Y                  [,1]      [,2]
##   dissatisfied 2.934567 0.5691214
##   satisfied    3.619831 0.5884733
```

Naive Bayes Model Summary

The Bayes model output shows us that depending on the predictors, we would be able to guess a certain amount of probability correct or incorrect. The A-priori probabilities show us that among the entire dataset, 45% are dissatisfied with the service and only 54% are satisfied. For the first conditional probability table, it shows us that predicting based on gender will show us that about 39% dissatisfied guests are female and 60% of them are male. The same can be stated with the rest of the tables.

Next, we can see a table that states what our model predicted and the actual outcome again. We can already see that the Naive Bayes model is better, since it predicted more satisfied and dissatisfied customers to what they actually were. '

```
p2 <- predict(nb1, newdata=sortByRating2Test, type = "class")
table(p2, test$satisfaction)
```

```
##
## p2             dissatisfied satisfied
##   dissatisfied         6691      4394
##   satisfied            5102      9789
```

The accuracy for Naive Bayes was over 63%, a huge improvement!

```
mean(p2==test$satisfaction)
```

```
## [1] 0.6344318
```

Evaluating both Models

Based on the 2 different types of models, the Bayes model was better suited for our data set because the prediction was around 64% accurate while the logistic regression model was only 34% accurate. I think the reason that the logistic regression model was not as accurate was because it only took into account the average rating to determine satisfaction while the Bayes model took in multiple predictors to determine the satisfaction.

Strengths and Weaknesses of Both Models

The strengths of Bayes is that it allows the user to take in more predictors to be able to produce a quick summary output that will be somewhat accurate. The Logistic regression strength is that it is better at predicting if the data being sent in has a trend or pattern that the machine can learn. The weakness of Bayes model is that because it is simple and fast at producing an output, other types of models will be better and more accurate if the data is better. The weakness of Logistic regression is that the data given to the model will ultimately determine the output. If bad data is sent in then the output will be inaccurate and if there is no relationship between the predictor and target then Logistic regression becomes useless.

Conclusion with Classification Metrics Used

The benefit of using Bayes is that it allows the machine to take in multiple predictors to be able to make a quick prediction and be somewhat accurate but the drawback is that because it is quick to produce, it is not always the best tool to use to make sure that the machine is able to predict correctly. The benefit of Logistic regression is that as long as there is a pattern between the predictors and the target, the logistic regression is great at predicting, however the downside is that if the data used to predict are irrelevant, the predictions will be unreliable.