"Airline Passenger Satisfaction Project"

Brian Clark

6/1/2021

Contents

1. General

1.1 Introduction

For this project, I used a dataset found on the Kaggle website (https://www.kaggle.com/datasets) called Airline Passenger Satisfaction. The data consists of a survey used to determine airline passenger satisfaction. The dataset is provided by TJ Klein, a Servo Firmware Engineer at Seagate Technology in Minneapolis, Minnesota. My goal is to explore the data to determine its usability and then apply a few machine learning algorithms in the caret package to determine which one most correctly identifies the satisfied airline passenger.

1.2 Overview

My data exploration centers around data clustering techniques and the application of Principal Components Analysis. I then apply three machine learning algorithms based on their ability to combine variables using machine learning techniques such as clustering, nearest neighbor, and random forest. I also create an Ensemble model using the top two best machine learning models. I base my final observations on the application of these models.

1.3 Executive Summary

For this project, I examined machine learning algorithms that go beyond standard linear and logistic regression. I applied those models to a dataset found on the Kaggle website called Airline Passenger Satisfaction. This dataset contains an airline passenger satisfaction survey. The data columns are as follows:

Column 01: X: Column Number
Column 02: id: Survey ID
Column 03: Gender: Gender of the passengers (Female, Male)
Column 04: Customer Type: The customer type (Loyal customer, disloyal customer)
Column 05: Type of Travel: Purpose of the flight (Personal Travel, Business Travel)
Column 06: Age: The actual age of the passengers
Column 07: Class: Travel class (Business, Eco, Eco Plus)
Column 08: Flight distance: The flight distance of this journey
Column 09: Inflight wifi service: Satisfaction level of wifi service (0:Not Applicable;1-5)
Column 10: Departure/Arrival time convenience: Satisfaction level (0:Not Applicable;1-5)
Column 11: Ease of Online booking: Satisfaction level of online booking (0:Not Applicable;1-5)
Column 12: Gate location: Satisfaction level of Gate location (0:Not Applicable;1-5)
Column 13: Food and drink: Satisfaction level of Food and drink (0:Not Applicable;1-5)
Column 14: Online boarding: Satisfaction level of online boarding (0:Not Applicable;1-5)
Column 15: Seat comfort: Satisfaction level of Seat comfort (0:Not Applicable;1-5)
Column 16: Inflight entertainment: Satisfaction level of entertainment (0:Not Applicable;1-5)
Column 17: On-board service: Satisfaction level of On-board service (0:Not Applicable;1-5)
Column 18: Leg room service: Satisfaction level of Leg room service (0:Not Applicable;1-5)
Column 19: Baggage handling: Satisfaction level of baggage handling (0:Not Applicable;1-5)
Column 20: Check-in service: Satisfaction level of Check-in service (0:Not Applicable;1-5)
Column 21: Inflight service: Satisfaction level of inflight service (0:Not Applicable;1-5)
Column 22: Cleanliness: Satisfaction level of Cleanliness (0:Not Applicable;1-5)
Column 23: Departure Delay in Minutes: Minutes delayed when departing
Column 24: Arrival Delay in Minutes: Minutes delayed when arriving
Column 25: Satisfaction: Airline satisfaction level (Satisfaction, neutral or dissatisfaction)

I began by removing the first two columns as they are unique identifiers. I then shortened the variable names and converted variables with character values to numeric. I did this because my analysis methodology requires numeric data.

Next, I reviewed the revised data and cleaned it up for subsequent analysis. I categorized age and flight distance and removed two columns with near zero variance. Finally, I identified the survey data with zero (Not Applicable) values and removed those rows from the dataset. The remaining data consisted of 23,863 rows and 21 columns. The last column, entitled "satisfaction", became my "y" values as this column contained the ultimate results of the survey.

My goal was to explore the data to determine its usability and then apply a few appropriate machine learning algorithms in the caret package to determine which one most correctly identifies the satisfied airline passenger.

A necessary step in my analysis was to scale the data in order to standardize the range of the initial variables so that each one of them contributed equally to the various analyses.

Because 20 columns and 23,863 rows contain a large amount of information, I then explored the data to determine whether or not I could use a few machine learning models based on their ability to combine variables and improve the modeling process. I started my exploration with hierarchical clustering and then applied Principal Components Analysis. Based on these observations, I concluded that the dataset is amenable to applying machine learning techniques that are based on their ability to combine variables using machine learning techniques such as clustering, nearest neighbor, and random forest.

Then, using the caret package, I applied three machine learning models: K-Means Clustering, K-Nearest Neighbors, and Random Forest. I also created an Ensemble model using the top two best machine learning algorithms. The top two models are the K-Nearest Neighbors and the Random Forest models. As a result of my analyses, I determined that application of the Random Forest model provides the most accuracy; however, the Ensemble model, close behind in accuracy, actually provides better sensitivity, or a better ability to correctly predict satisfied customers. Here is a summary of my results:

Model	Accuracy	Sensitivity	Specificity
K Means	0.7894	0.8614	0.7350
K Nearest Neighbors	0.9337	0.8953	0.9628
Random Forest	0.9583	0.9408	0.9716
Ensemble	0.9516	0.9496	0.9531

2. Methods and Analysis

My project goal is to explore and analyze TJ Klein's Airline Passenger Satisfaction survey as found on the Kaggle website and then apply a few appropriate machine learning algorithms in the caret package to determine which one most correctly identifies the satisfied airline passenger. Due to the massive size of this database, I chose to work with a smaller subset also provided on the same website. I began the project by observing the nature of the data and cleaning it to remove unneeded information and to shorten unwieldy column names. Then I used various techniques to explore and visualize the data. In the modeling portion of the project, I apply four machine learning algorithms. The caret package provides a convenient means for accomplishing this and allows me to determine which model most correctly identifies the satisfied airline passenger.

2.1 Data Initialization and Cleaning

Load Libraries and Specify Rounding

I begin my project by loading the following packages:

- tidyverse – makes it easy to install and load core packages in a single command.
- caret – contains functions to streamline the model training process.

- matrixStats - high-performing functions operating on rows and columns of matrices.

I include "!require(package)" coding because the r code is shared for grading purposes. When loading a program that someone else is going to run, it's best to make sure the packages are installed.

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

library(tidyverse)
library(caret)
library(data.table)
```

I report all numeric results to four significant digits as a personal preference.

```
options(digits = 4)
```

Read Data

Airline Passenger Satisfaction The dataset for this project is found on Kaggle at

https://www.kaggle.com/teejmahal20/airline-passenger-satisfaction/download

Dataset Acknowledgement:

TJ Klein
Servo Firmware Engineer at Seagate Technology
Minneapolis, Minnesota, United States

The actual dataset is divided into two datasets named test.csv and train.csv. The combined dataset is extremely large and cumbersome for machine learning. So, for this assignment, I use just the smaller test.csv dataset and create my own test and train datasets from test.csv.

I have downloaded the test.csv dataset to my github repository and made the dataset public. Here is the web link for this dataset:

https://raw.githubusercontent.com/btclark34/Create_Your_Own_BClark/main/test.csv

And, here is how I read the file.

```
rddata <- "https://raw.githubusercontent.com/btclark34/Create_Your_Own_BClark/main/test.csv"
rd <- read.csv(rddata, header = TRUE)
```

Here is a summary of the data columns.

Column 01: X: Column Number
Column 02: id: Survey ID
Column 03: Gender: Gender of the passengers (Female, Male)
Column 04: Customer Type: The customer type (Loyal customer, disloyal customer)
Column 05: Type of Travel: Purpose of the flight (Personal Travel, Business Travel)
Column 06: Age: The actual age of the passengers
Column 07: Class: Travel class (Business, Eco, Eco Plus)
Column 08: Flight distance: The flight distance of this journey
Column 09: Inflight wifi service: Satisfaction level of wifi service (0:Not Applicable;1-5)
Column 10: Departure/Arrival time convenience: Satisfaction level (0:Not Applicable;1-5)
Column 11: Ease of Online booking: Satisfaction level of online booking (0:Not Applicable;1-5)
Column 12: Gate location: Satisfaction level of Gate location (0:Not Applicable;1-5)

Column 13: Food and drink: Satisfaction level of Food and drink (0:Not Applicable;1-5)
Column 14: Online boarding: Satisfaction level of online boarding (0:Not Applicable;1-5)
Column 15: Seat comfort: Satisfaction level of Seat comfort (0:Not Applicable;1-5)
Column 16: Inflight entertainment: Satisfaction level of entertainment (0:Not Applicable;1-5)
Column 17: On-board service: Satisfaction level of On-board service (0:Not Applicable;1-5)
Column 18: Leg room service: Satisfaction level of Leg room service (0:Not Applicable;1-5)
Column 19: Baggage handling: Satisfaction level of baggage handling (0:Not Applicable;1-5)
Column 20: Check-in service: Satisfaction level of Check-in service (0:Not Applicable;1-5)
Column 21: Inflight service: Satisfaction level of inflight service (0:Not Applicable;1-5)
Column 22: Cleanliness: Satisfaction level of Cleanliness (0:Not Applicable;1-5)
Column 23: Departure Delay in Minutes: Minutes delayed when departing
Column 24: Arrival Delay in Minutes: Minutes delayed when arriving
Column 25: Satisfaction: Airline satisfaction level (Satisfaction, neutral or dissatisfaction)

I would like to know, up front and before I start manipulating the data, what proportion of the airline customers in the original dataset are identified as satisfied customers.

```
mean(rd$satisfaction == "satisfied")
```

```
[1] 0.439
```

This does not seem like a very high proportion of satisfied customers. I can see a need to determine what will satisfy airline customers.

Data Cleaning

Here are the dimensions of the dataset.

```
dim(rd)
```

```
[1] 25976    25
```

Now, let's take a brief look at how the dataset we just read is set up.

```
head(rd,3)
```

```
  X    id Gender     Customer.Type Age  Type.of.Travel    Class Flight.Distance
1 0 19556 Female    Loyal Customer  52 Business travel      Eco             160
2 1 90035 Female    Loyal Customer  36 Business travel Business            2863
3 2 12360   Male disloyal Customer  20 Business travel      Eco             192
  Inflight.wifi.service Departure.Arrival.time.convenient
1                     5                                4
2                     1                                1
3                     2                                0
  Ease.of.Online.booking Gate.location Food.and.drink Online.boarding
1                      3             4              3               4
2                      3             1              5               4
3                      2             4              2               2
  Seat.comfort Inflight.entertainment On.board.service Leg.room.service
1            3                      5                5                5
2            5                      4                4                4
3            2                      2                4                1
```

```
  Baggage.handling Checkin.service Inflight.service Cleanliness
1                5               2                5           5
2                4               3                4           5
3                3               2                2           2
  Departure.Delay.in.Minutes Arrival.Delay.in.Minutes            satisfaction
1                         50                       44               satisfied
2                          0                        0               satisfied
3                          0                        0 neutral or dissatisfied
```

The first two columns are unique identifiers and not necessary for my analysis, so I will remove them.

```
rd <- rd[, c(3:25)]
```

The column names are rather lengthy. I would like to simplify the coding and data visualization, so I am going to rename each of them to their column number.

```
colnames(rd) <- c(3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25)
```

The analysis portion of my project begins with matrix scaling which is used throughout the rest of the project. Matrix scaling requires numeric data. So, I change character column values to numeric values.

```
rd$`3`[rd$`3` == "Female"] <- 1
rd$`3`[rd$`3` == "Male"] <- 2
rd$`3` <- as.integer(rd$`3`)

rd$`4`[rd$`4` == "Loyal Customer"] <- 1
rd$`4`[rd$`4` == "disloyal Customer"] <- 2
rd$`4` <- as.integer(rd$`4`)

rd$`6`[rd$`6` == "Business travel"] <- 1
rd$`6`[rd$`6` == "Personal Travel"] <- 2
rd$`6` <- as.integer(rd$`6`)

rd$`7`[rd$`7` == "Business"] <- 1
rd$`7`[rd$`7` == "Eco Plus"] <- 2
rd$`7`[rd$`7` == "Eco"] <- 3
rd$`7` <- as.integer(rd$`7`)
```

I also change the "y" values to numeric as a matter of convenience.

```
rd$`25`[rd$`25` == "neutral or dissatisfied"] <- 0
rd$`25`[rd$`25` == "satisfied"] <- 1
rd$`25` <- as.integer(rd$`25`)
```

Now let's look at a summary of the new dataset.

```
summary(rd)
```

```
       3               4               5               6               7
 Min.   :1.00    Min.   :1.00    Min.   : 7.0    Min.   :1.00    Min.   :1.00
```

```
1st Qu.:1.00   1st Qu.:1.00   1st Qu.:27.0   1st Qu.:1.00   1st Qu.:1.00
Median :1.00   Median :1.00   Median :40.0   Median :1.00   Median :2.00
Mean   :1.49   Mean   :1.18   Mean   :39.6   Mean   :1.31   Mean   :1.96
3rd Qu.:2.00   3rd Qu.:1.00   3rd Qu.:51.0   3rd Qu.:2.00   3rd Qu.:3.00
Max.   :2.00   Max.   :2.00   Max.   :85.0   Max.   :2.00   Max.   :3.00

      8              9              10             11             12
Min.   :  31   Min.   :0.00   Min.   :0.00   Min.   :0.00   Min.   :1.00
1st Qu.: 414   1st Qu.:2.00   1st Qu.:2.00   1st Qu.:2.00   1st Qu.:2.00
Median : 849   Median :3.00   Median :3.00   Median :3.00   Median :3.00
Mean   :1194   Mean   :2.73   Mean   :3.05   Mean   :2.76   Mean   :2.98
3rd Qu.:1744   3rd Qu.:4.00   3rd Qu.:4.00   3rd Qu.:4.00   3rd Qu.:4.00
Max.   :4983   Max.   :5.00   Max.   :5.00   Max.   :5.00   Max.   :5.00

      13             14             15             16             17
Min.   :0.00   Min.   :0.00   Min.   :1.00   Min.   :0.00   Min.   :0.00
1st Qu.:2.00   1st Qu.:2.00   1st Qu.:2.00   1st Qu.:2.00   1st Qu.:2.00
Median :3.00   Median :4.00   Median :4.00   Median :4.00   Median :4.00
Mean   :3.21   Mean   :3.26   Mean   :3.45   Mean   :3.36   Mean   :3.39
3rd Qu.:4.00   3rd Qu.:4.00   3rd Qu.:5.00   3rd Qu.:4.00   3rd Qu.:4.00
Max.   :5.00   Max.   :5.00   Max.   :5.00   Max.   :5.00   Max.   :5.00

      18             19             20             21             22
Min.   :0.00   Min.   :1.00   Min.   :1.00   Min.   :0.00   Min.   :0.00
1st Qu.:2.00   1st Qu.:3.00   1st Qu.:3.00   1st Qu.:3.00   1st Qu.:2.00
Median :4.00   Median :4.00   Median :3.00   Median :4.00   Median :3.00
Mean   :3.35   Mean   :3.63   Mean   :3.31   Mean   :3.65   Mean   :3.29
3rd Qu.:4.00   3rd Qu.:5.00   3rd Qu.:4.00   3rd Qu.:5.00   3rd Qu.:4.00
Max.   :5.00   Max.   :5.00   Max.   :5.00   Max.   :5.00   Max.   :5.00

      23              24              25
Min.   :   0.0   Min.   :   0.0   Min.   :0.000
1st Qu.:   0.0   1st Qu.:   0.0   1st Qu.:0.000
Median :   0.0   Median :   0.0   Median :0.000
Mean   :  14.3   Mean   :  14.7   Mean   :0.439
3rd Qu.:  12.0   3rd Qu.:  13.0   3rd Qu.:1.000
Max.   :1128.0   Max.   :1115.0   Max.   :1.000
                 NA's   :83
```

Based on this summary, for more meaningful results, I would like to categorize age as follows:

```
rd$`5`[rd$`5` <= 14] <- 1                  # Ages 07-14 Children
rd$`5`[rd$`5` > 14 & rd$`5` <= 24] <- 2    # Ages 15-24 Youth
rd$`5`[rd$`5` > 24 & rd$`5` <= 64] <- 3    # Ages 25-64 Adults
rd$`5`[rd$`5` > 64] <- 4                   # Ages 65+   Seniors
rd$`5` <- as.integer(rd$`5`)
```

and flight distance as follows:

```
rd$`8`[rd$`8` <= 700] <- 1                  # Miles <= 700          Short
rd$`8`[rd$`8` >700 & rd$`8` < 2400] <- 2    # Miles > 700 & < 2400 Medium
rd$`8`[ rd$`8` >= 2400] <- 3                # Miles >= 2400         Long
rd$`8` <- as.integer(rd$`8`)
```

I also suspect that columns 23 and 24 have little to offer. Let's see.

```
nzv <- nearZeroVar(rd)
nzv
```

```
[1] 21 22
```

As suspected, columns 21 (named 23) and 22 (named 24) have near zero variance and will contribute little to our machine learning. I will remove both items.

```
rd <- rd[, c(1:20,23)]
```

I would also like to identify zeros or "Not Applicable" on the survey responses (columns 9-22) as "NA".

```
rd$`9`[ rd$`9` == 0] <- NA
rd$`10`[ rd$`10` == 0] <- NA
rd$`11`[ rd$`11` == 0] <- NA
rd$`12`[ rd$`12` == 0] <- NA
rd$`13`[ rd$`13` == 0] <- NA
rd$`14`[ rd$`14` == 0] <- NA
rd$`15`[ rd$`15` == 0] <- NA
rd$`16`[ rd$`16` == 0] <- NA
rd$`17`[ rd$`17` == 0] <- NA
rd$`18`[ rd$`18` == 0] <- NA
rd$`19`[ rd$`19` == 0] <- NA
rd$`20`[ rd$`20` == 0] <- NA
rd$`21`[ rd$`21` == 0] <- NA
rd$`22`[ rd$`22` == 0] <- NA
```

I would like to remove the rows with NA's. This will have little impact on the overall results because the dataset is so large.

```
rd <- na.omit(rd)
```

2.2 Data Exploration and Visualization

Now let's look at the refined dataset. Here are the dimensions.

```
dim(rd)
```

```
[1] 23863    21
```

As you can see, even though I used a subset of the original dataset, this dataset is still quite large.

Here is the overall summary of the refined data.

```
summary(rd)
```

```
       3               4               5               6               7        
 Min.   :1.00   Min.   :1.00   Min.   :1.00   Min.   :1.00   Min.   :1.00  
 1st Qu.:1.00   1st Qu.:1.00   1st Qu.:3.00   1st Qu.:1.00   1st Qu.:1.00  
 Median :1.00   Median :1.00   Median :3.00   Median :1.00   Median :2.00  
 Mean   :1.49   Mean   :1.16   Mean   :2.83   Mean   :1.31   Mean   :1.95  
 3rd Qu.:2.00   3rd Qu.:1.00   3rd Qu.:3.00   3rd Qu.:2.00   3rd Qu.:3.00  
 Max.   :2.00   Max.   :2.00   Max.   :4.00   Max.   :2.00   Max.   :3.00  
       8               9              10             11             12        
 Min.   :1.00   Min.   :1.00   Min.   :1.0   Min.   :1.00   Min.   :1.00  
 1st Qu.:1.00   1st Qu.:2.00   1st Qu.:2.0   1st Qu.:2.00   1st Qu.:2.00  
 Median :2.00   Median :3.00   Median :3.0   Median :3.00   Median :3.00  
 Mean   :1.75   Mean   :2.81   Mean   :3.2   Mean   :2.88   Mean   :2.99  
 3rd Qu.:2.00   3rd Qu.:4.00   3rd Qu.:4.0   3rd Qu.:4.00   3rd Qu.:4.00  
 Max.   :3.00   Max.   :5.00   Max.   :5.0   Max.   :5.00   Max.   :5.00  
       13              14              15              16              17        
 Min.   :1.00   Min.   :1.00   Min.   :1.00   Min.   :1.00   Min.   :1.00  
 1st Qu.:2.00   1st Qu.:2.00   1st Qu.:2.00   1st Qu.:2.00   1st Qu.:2.00  
 Median :3.00   Median :4.00   Median :4.00   Median :4.00   Median :4.00  
 Mean   :3.22   Mean   :3.35   Mean   :3.46   Mean   :3.38   Mean   :3.39  
 3rd Qu.:4.00   3rd Qu.:4.00   3rd Qu.:5.00   3rd Qu.:4.00   3rd Qu.:4.00  
 Max.   :5.00   Max.   :5.00   Max.   :5.00   Max.   :5.00   Max.   :5.00  
       18              19              20              21              22        
 Min.   :1.00   Min.   :1.00   Min.   :1.00   Min.   :1.00   Min.   :1.00  
 1st Qu.:2.00   1st Qu.:3.00   1st Qu.:3.00   1st Qu.:3.00   1st Qu.:2.00  
 Median :4.00   Median :4.00   Median :3.00   Median :4.00   Median :3.00  
 Mean   :3.38   Mean   :3.64   Mean   :3.31   Mean   :3.65   Mean   :3.29  
 3rd Qu.:4.00   3rd Qu.:5.00   3rd Qu.:4.00   3rd Qu.:5.00   3rd Qu.:4.00  
 Max.   :5.00   Max.   :5.00   Max.   :5.00   Max.   :5.00   Max.   :5.00  
       25        
 Min.   :0.00  
 1st Qu.:0.00  
 Median :0.00  
 Mean   :0.43  
 3rd Qu.:1.00  
 Max.   :1.00  
```

Now let's look at the proportion of satisfied customers.

```
mean(rd$`25` == 1)
```

```
[1] 0.4305
```

The refined dataset maintains approximately the same proportion of satisfied customers as my original dataset. This really comes as no surprise.

Next, I create a new dataset to aid in my exploration and modeling of the data. I begin by creating a null dataset so that I can create x and y subsets within that dataset.

```
sat <- list()
```

I create the "x" values based on the 20 columns remaining from the original rd dataset. There are actually 21 columns remaining from the orignal rd; however, the last column contains the y values.

sat$x is created as a matrix as required for subsequent analysis.

```
sat$x <- as.matrix(rd[, c(1:20)])
```

sat$y is created as a factor as required for subsequent analysis.

```
sat$y <- factor(rd$`25`)
```

Now, let's check the dimensions of sat$x and match to the original rd dataset.

```
dim(sat$x)
```

```
[1] 23863    20
```

What proportion of the passengers are satisfied customers? Does it match the refined rd dataset?

```
mean(sat$y == 1)
```

```
[1] 0.4305
```

As expected, the proportion of satisfied customers is the same as my refined dataset.

Matrix Scaling

Now we scale the "x" matrix in order to standardize the "x" values. We scale the matrix so that one column variable doesn't overly impact the model simply because of its magnitude. So, the goal of this step is to standardize the range of the initial variables so that each one of them contributes equally to the analysis.

The scaling operation is performed using the sweep function twice. First, we take the column means and subtract them from the corresponding column values. Then we divide the resulting matrix values by the standard deviation.

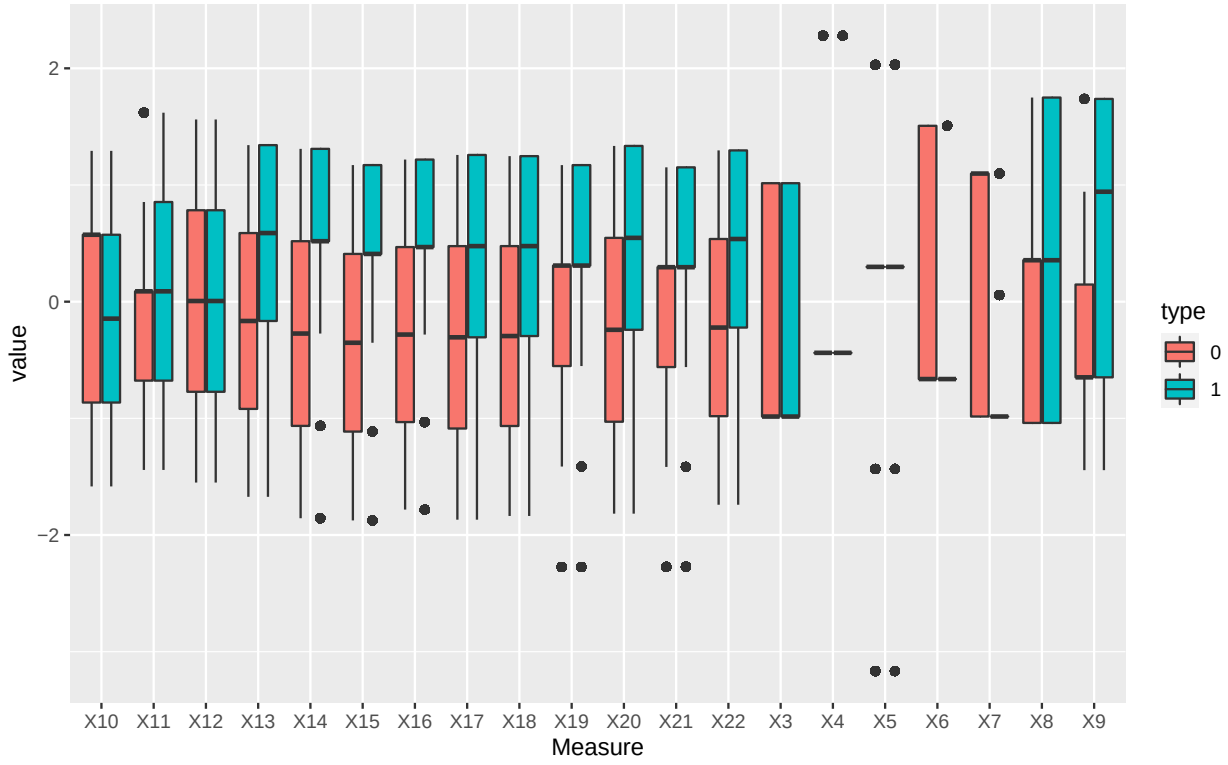
Once the values are scaled or standardized, the mean of these standardized values is always zero, and the standard deviation is always one.

Matrix scaling in machine learning is one of the most critical steps during the pre-processing of data before creating a machine learning model. Scaling can make a difference between a weak machine learning model and a better one.

```
x_centered <- sweep(sat$x, 2, colMeans(sat$x))
x_scaled <- sweep(x_centered, 2, colSds(sat$x), FUN = "/")
```

Let's see the scaled matrix from a graphical perspective.

```
data.frame(type = sat$y, x_scaled[,1:20]) %>%
gather(key = "Measure", value = "value", -type) %>%
ggplot(aes(Measure, value, fill = type)) +
geom_boxplot()
```



Clustering

Clustering is a machine learning technique in which we do not necessarily know the outcomes, but, instead, are interested in discovering groups or clusters. Because we have so many variables in this dataset, I am interested in the potential for combining similar variables while applying various modeling techniques. Clustering will help me to see this potential.

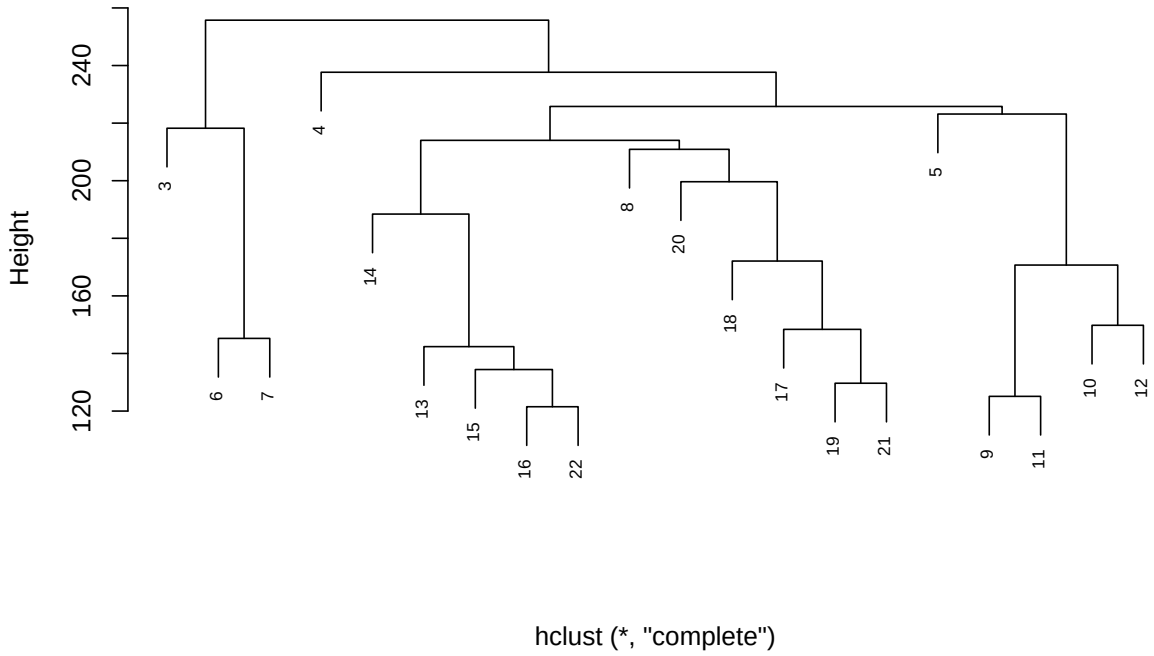
Hierarchical Clustering

We need an algorithm to define groups from the scaled matrix. Hierarchical clustering starts by defining each observation as a separate group, then the two closest groups are joined into a group iteratively until there is just one group including all the observations. The hclust function implements this algorithm and allows us to define groups or clusters.

```
d_features <- dist(t(x_scaled))
h <-hclust(d_features)
```

Let's see what the resulting groups look like in a dendogram.

```
plot(h, cex = 0.65, main = "", xlab = "")
```



hclust (*, "complete")

To generate actual groups, we can decide on the number of groups we want and then find the minimum distance that achieves this. I base the number of groups (k) on the dendogram. This is subjective, but I can divide the dendogram into 7 separate groups

```
groups <- cutree(h, k=7)
split(names(groups), groups)
```

```
$`1`
[1] "3"

$`2`
[1] "4"

$`3`
[1] "5"

$`4`
[1] "6" "7"

$`5`
[1] "8"  "17" "18" "19" "20" "21"

$`6`
[1] "9"  "10" "11" "12"

$`7`
[1] "13" "14" "15" "16" "22"
```

What we see here is:

Group 1 is based on Gender.
Group 2 is based on Customer Type (Loyal or Disloyal).
Group 3 is Type of Travel (Personal or Business). Group 4 is based on Age and Class (Business, Economy, or Economy Plus). Group 5 is Flight Distance, On-Board Service, Leg Room Service, Baggage Handling, Check-in Service, and Inflight Service which tend to deal with how the customer is treated while flying.
Group 6 is Inflight wifi service, Convenience of Departure/Arrival Time, Ease of Online Booking and Gate Location which actually seem like two separate issues; On-line services and how the customer is treated at departure and arrival.
Group 7 is Food and Drink, Online Boarding, Seat Comfort, Inflight Entertainment, and Cleanliness all of which relate to comfort while flying.

In other words, we tend see some similarities within some of these cluster characteristics. This may allow us to combine variables and reduce the overall number when applying various modeling techniques. But first, let's review the data a little further.

Principal Component Analysis

The scaled matrix is also important here. It is critical to perform standardization prior to Principal Components Analysis (PCA) because PCA is very sensitive to the variances of the initial variables. In other words, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges. For example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1, which will lead to biased results. So, transforming the data to comparable scales can help to prevent this problem.

Principal Component Analysis (PCA) is an exploratory tool for data analysis. The idea behind PCA is simple—reduce the number of variables of a data set, while preserving as much information as possible. Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the idea in dimensionality reduction is to trade a little accuracy for simplicity.

As there are as many principal components as there are variables in the data, principal components are constructed in such a manner that the first principal component accounts for the largest possible variance in the data set. The second principal component is calculated in the same way, with the condition that it is uncorrelated with (or perpendicular to) the first principal component and that it accounts for the next highest variance. This continues until all principal components have been calculated, equal to the original number of variables.

Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (the principal components) are uncorrelated and most of the information within the initial variables is squeezed or compressed into the first components. So, the idea is 20-dimensional data gives you 20 principal components, but PCA tries to put maximum possible information in the first component, then maximum remaining information in the second and so on, until we have something as shown in the summary below.

```
pca <- prcomp(x_scaled)
summary(pca)
```

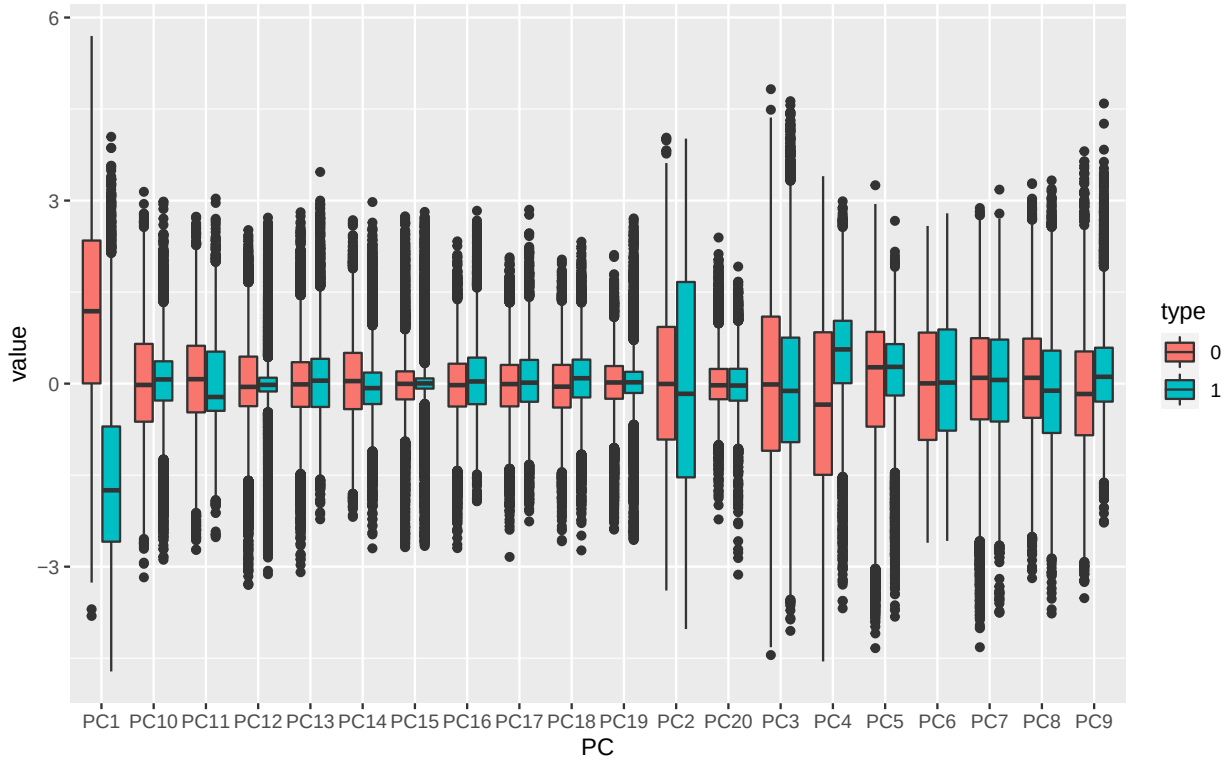
```
Importance of components:
                          PC1   PC2   PC3    PC4    PC5    PC6    PC7    PC8
Standard deviation     2.040 1.586 1.482 1.3274 1.1393 1.0085 0.9807 0.9752
Proportion of Variance 0.208 0.126 0.110 0.0881 0.0649 0.0508 0.0481 0.0476
Cumulative Proportion  0.208 0.334 0.444 0.5317 0.5966 0.6474 0.6955 0.7430
                          PC9   PC10   PC11   PC12   PC13   PC14   PC15   PC16
Standard deviation     0.9239 0.8227 0.7782 0.6831 0.6619 0.6383 0.5913 0.5698
```

```
Proportion of Variance 0.0427 0.0338 0.0303 0.0233 0.0219 0.0204 0.0175 0.0162
Cumulative Proportion  0.7857 0.8196 0.8499 0.8732 0.8951 0.9155 0.9329 0.9492
                          PC17   PC18   PC19    PC20
Standard deviation     0.5401 0.5302 0.5170 0.41990
Proportion of Variance 0.0146 0.0141 0.0134 0.00882
Cumulative Proportion  0.9638 0.9778 0.9912 1.00000
```

An important thing to realize here is that, the principal components are less interpretable and don't have any real meaning since they are constructed as linear combinations of the initial variables.

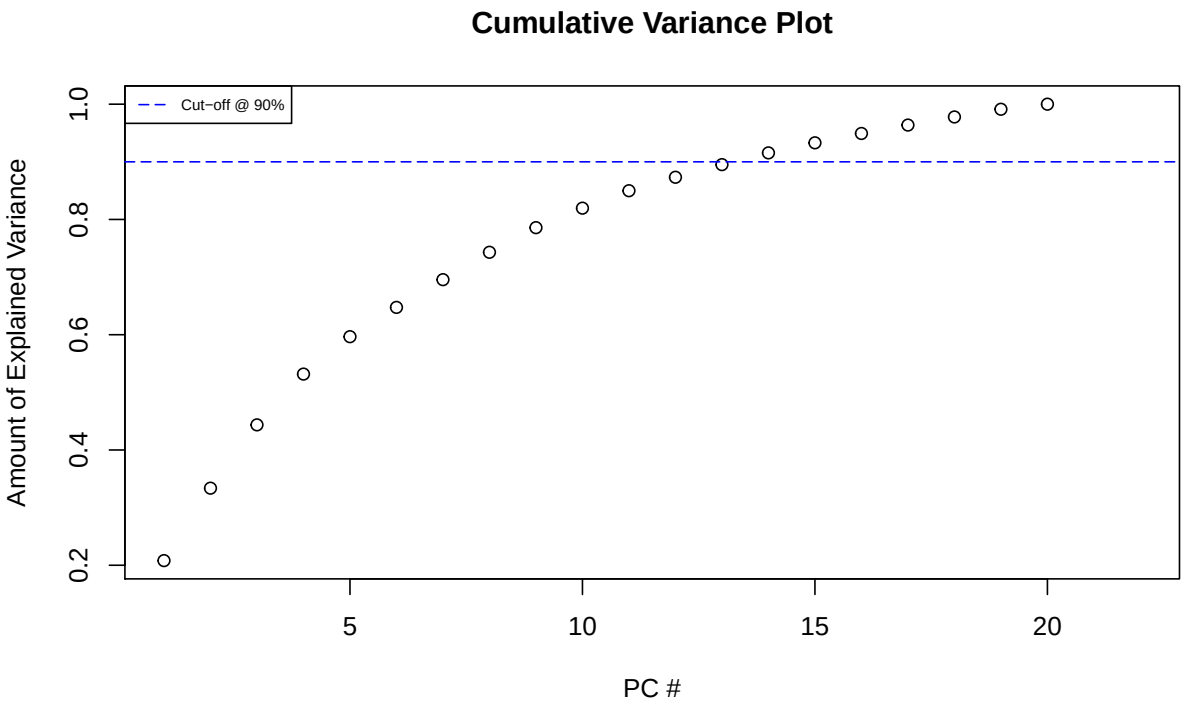
Let's see the Prinicipal Components from a graphical perspective.

```
data.frame(type = sat$y, pca$x[,1:20]) %>%
gather(key = "PC", value = "value", -type) %>%
ggplot(aes(PC, value, fill = type)) +
geom_boxplot()
```



Now, let's look at PCA again using a table of the Cumulative Variance plotted against the Amount of Explained Variance.

```
cumpro <- cumsum(pca$sdev^2 / sum(pca$sdev^2))
plot(cumpro[0:22], xlab = "PC #", ylab = "Amount of Explained Variance", main = "Cumulative Variance Pl
abline(h = 0.9, col="blue", lty=5)
legend("topleft", legend=c("Cut-off @ 90%"), col=c("blue"), lty=5, cex=0.6)
```



I use 90% as the cutoff value to identify how many Principal Components we need to achieve 90% accuracy. It takes 13 principal components to attain 90% accuracy. This is still a large number of variables, but better than 20.

What this tells me is that based on my data exploration and visualization, the dataset appears amenable to combining variables using machine learning techniques such as clustering, nearest neighbor, and random forest type models. Let's explore these modeling techniques.

2.3 Modeling

Build Training and Test Sets

First, we need to create the training and test datasets by splitting sat$y and the scaled version of the sat x matrix (x_scaled) into 50% test set and 50% train set. Typically, I might split the data into 80% training and 20% testing (or even 75% training and 25% testing) so that I would have the bulk of the data for model training. For this analysis, I split the data half and half between training and test datasets because the overall dataset is so large and for some models this will reduce the training timeframe while still maintaining sufficient data for an accurate analysis.

Now let's create the test datasets.

```
set.seed(143, sample.kind = "Rounding")    # if using R 3.6 or later
test_index <- createDataPartition(sat$y, times = 1, p = 0.5, list = FALSE)
test_x <- x_scaled[test_index,]
test_y <- sat$y[test_index]
```

How many rows and columns does test_x have?

```
dim(test_x)
```

```
[1] 11932    20
```

We have the same number of columns as x_scaled, which is what we wanted, and 50% of the rows in x_scaled which is also what we wanted.

How many rows does test_y have?

```
dim(data.frame(test_y))
```

```
[1] 11932     1
```

We have the same number of rows as test_x which is what we needed.

Now let's create the training datasets.

```
train_x <- x_scaled[-test_index,]
train_y <- sat$y[-test_index]
```

How many rows and columns does train_x have? Does the number of columns agree with test_x?

```
dim(train_x)
```

```
[1] 11931    20
```

train_x has the same number of columns as test_x, which is what we wanted, and 50% of the rows in x_scaled which is also what we wanted.

How many rows does train_y have?

```
dim(data.frame(train_y))
```

```
[1] 11931     1
```

train_y has 50% of the rows of sat$y which is what we needed.

Let's check to make sure that the training and test sets have similar proportions of the satisfaction rating.

```
mean(train_y == 1)
```

```
[1] 0.4305
```

```
mean(test_y == 1)
```

```
[1] 0.4305
```

They are the same. We are now ready to move on to use some modeling techniques.

Modeling Approach

Now we'll look at various machine learning techniques using the caret package and determine which technique most accurately identifies the satisfied customer.

My final analysis is based on overall accuracy, sensitivity and specificity. The overall accuracy is simply defined as the overall proportion that is predicted correctly. In general, sensitivity is defined as the ability of an algorithm to predict a positive outcome when the actual outcome is positive: $\hat{Y} = 1$ when $Y = 1$. Because an algorithm that calls everything positive ($\hat{Y} = 1$ no matter what) has perfect sensitivity, this metric on its own is not enough to judge an algorithm. For this reason, I also examine specificity, which is generally defined as the ability of an algorithm to not predict a positive outcome $\hat{Y} = 0$ when the actual outcome is not a positive $Y = 0$.

K-Means Clustering

K-Means clustering works by partitioning n objects into k clusters in which each object belongs to the cluster with the nearest mean. This method produces exactly k different clusters of greatest possible distinction. The objective of K-Means clustering is to minimize total variance within each cluster.

To use the k-means clustering algorithm we have to pre-define k, the number of clusters or centers. The number of clusters I use here is based on repeating the algorithm until I find the cluster with the greatest accuracy. I only show that particular iteration here.

Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares. In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.

```
predict_kmeans <- function(x, k) {centers <- k$centers
distances <- sapply(1:nrow(x), function(i)
{apply(centers, 1, function(y) dist (rbind(x[i,], y)))})
max.col(-t(distances))}

set.seed(693, sample.kind = "Rounding")    # if using R 3.6 or later
k <- kmeans(train_x, centers = 2)
kmeans_preds <- ifelse(predict_kmeans(test_x, k) == 1, 0, 1)
mean(kmeans_preds == test_y)
```

```
[1] 0.7894
```

```
sensitivity(factor(kmeans_preds), test_y, positive = 1) # satisfied customer
```

```
[1] 0.8614
```

```
specificity(factor(kmeans_preds), test_y, negative = 0) # not satisfied or neutral customer
```

```
[1] 0.735
```

This is not all that accurate overall (only 79%). It is much better at predicting the satisfied customer (86%), but it's done at the expense of properly predicting the not satisfied or neutral customer (74%). We can do better than this.

K-Nearest Neighbors (Knn) Model

K-Nearest Neighbors algorithm, or Knn, is an approach to data classification that estimates how likely a data point is to belong to one group or another depending on where the data points nearest to it belong.

The primary assumption that a Knn model makes is that data points which exist in close proximity to each other are highly similar, while if a data point is far away from another group it's dissimilar to those data points.

A Knn model calculates similarity using the distance between two points. The greater the distance between the points, the less similar they are.

Again, we have to pre-define the k values. So, I ran some preliminary iterations of the model using different k values until I found the best. I only show that particular iteration here.

We can fit the Knn model with the caret function knn.

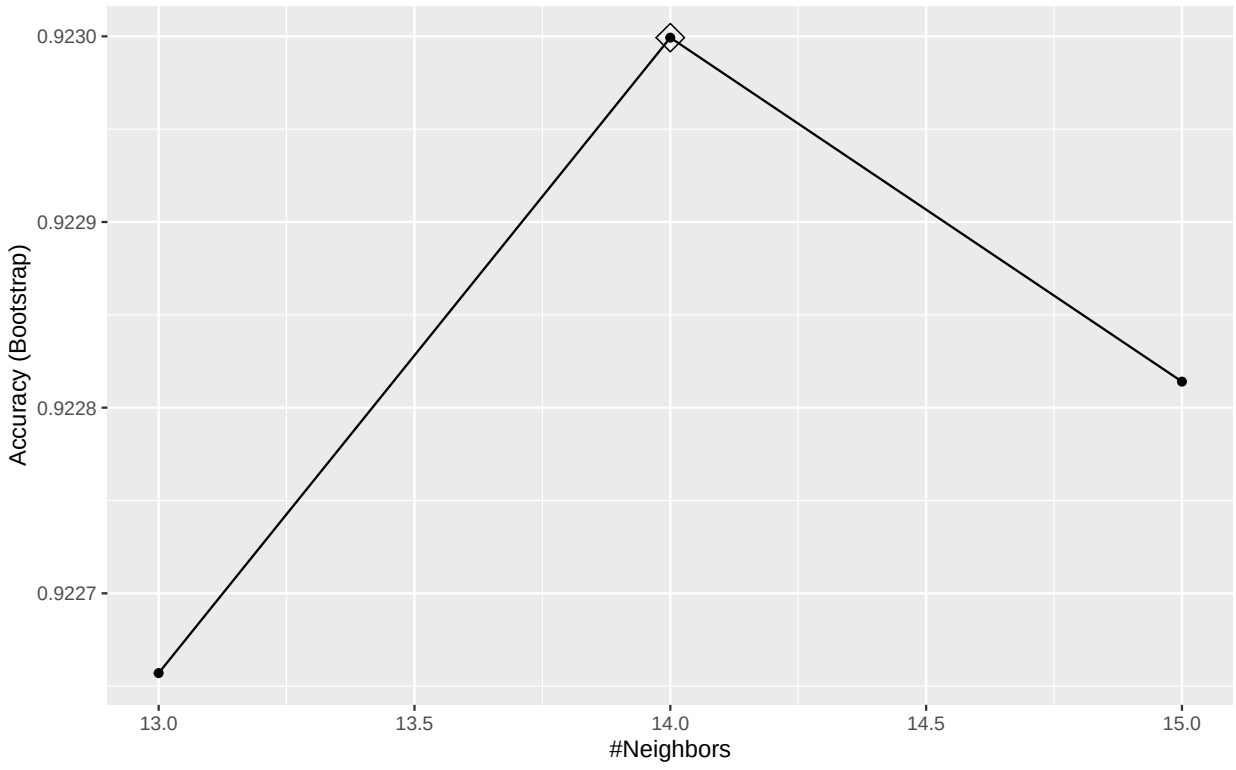
```
set.seed(447, sample.kind = "Rounding") # simulate R 3.5
tuning <- data.frame(k = seq(13, 15, 1))
train_knn <- train(train_x, train_y, method = "knn", tuneGrid = tuning)
```

What is the best k value for this model?

```
train_knn$bestTune
```

```
   k
2 14
```

```
ggplot(train_knn, highlight = TRUE)
```



Here we see that the best k value is 14.

Now let's look at the model accuracy.

```
train_knn$finalModel
```

```
14-nearest neighbor model
Training set outcome distribution:

   0    1
6795 5136
```

```
knn_preds <- predict(train_knn,test_x)
mean(knn_preds == test_y)
```

```
[1] 0.9337
```

The accuracy of the K-Nearest Neighbor model is 93%; much better than the K-Means Clustering.

```
sensitivity(factor(knn_preds), test_y, positive = 1) # satisfied customer
```

```
[1] 0.8953
```

```
specificity(factor(knn_preds), test_y, negative = 0) # not satisfied or neutral customer
```

```
[1] 0.9628
```

Sensitivity and specificity improved as well; however, the ability to correctly predict satisfied customers (sensitivity) only improved slightly (90%). The ability to correctly predict not satisfied or neutral customers (specificity) is much better at 96%.

Now let's see if the Random Forest Model can do even better.

Random Forest Model

Random forests are a learning method for classification and regression. They operate by constructing a multitude of decision trees at training time and outputting the class that is the mean prediction of the individual trees. Random forests improve prediction and reduce instability by averaging multiple decision trees (a forest of trees constructed with randomness).

With random forest, computation time is a challenge. For each forest, we need to build hundreds of trees. The coding provided below takes approximately 30 minutes to run.

Once again, I ran several iterations of this model until I found the optimum mtry value. mtry refers to the number of variables randomly sampled as candidates at each split. I only show the final iteration here.

A disadvantage of random forests is that we lose interpretability. An approach that helps with interpretability is to examine variable importance. To define variable importance, we count how often a predictor is used in the individual trees. The caret package includes the function varImp that extracts variable importance from any model in which the calculation is implemented.

We can fit the random forest model with the caret function rf.

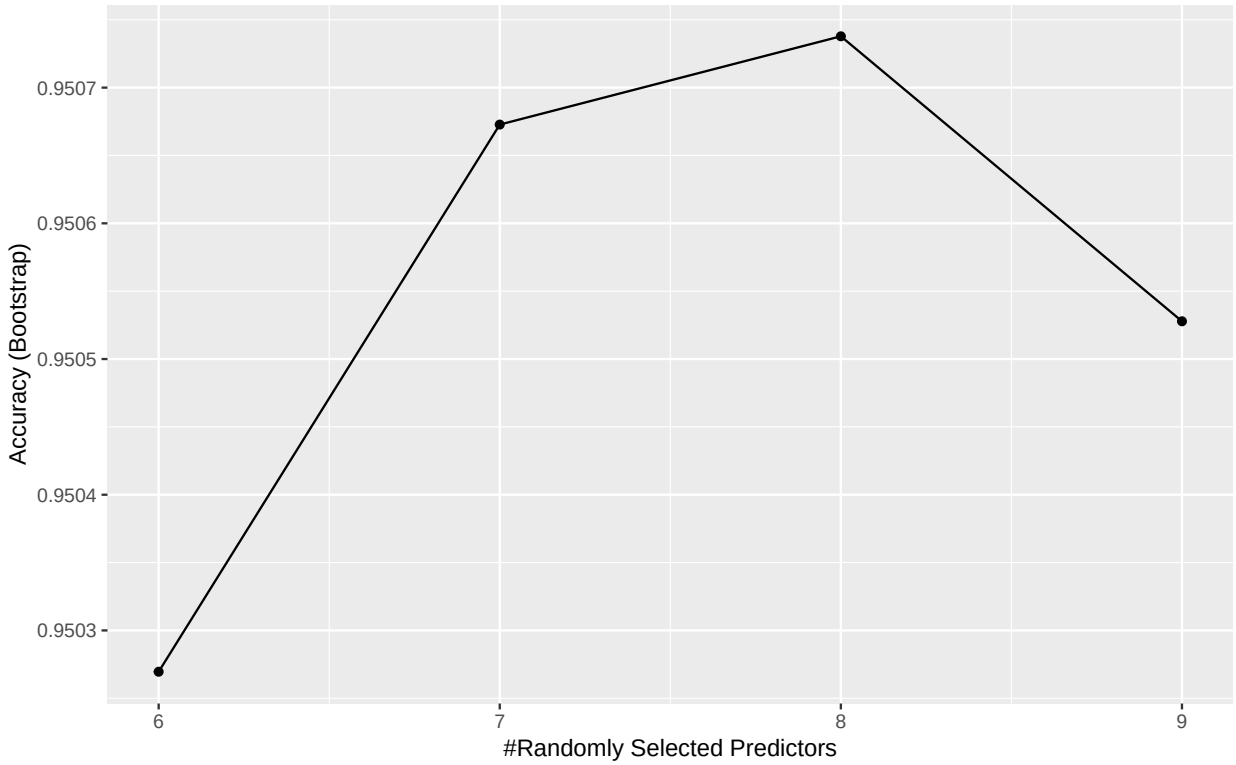
```
set.seed(901, sample.kind = "Rounding")
tuning <- data.frame(mtry = c(6, 7, 8, 9))
train_rf <- train(train_x, train_y,
method = "rf",
tuneGrid = tuning,
importance = TRUE)
```

What is the optimum mtry value?

```
train_rf$bestTune
```

```
  mtry
3    8
```

```
ggplot(train_rf)
```



We see that the optimum mtry value is 8.

Now let's look at the predictive value of this model.

```
rf_preds <- predict(train_rf,test_x)
mean(rf_preds == test_y)
```

```
[1] 0.9583
```

```
sensitivity(factor(rf_preds), test_y, positive = 1) # satisfied customer
```

```
[1] 0.9408
```

```
specificity(factor(rf_preds), test_y, negative = 0) # not satisfied or neutral customer
```

```
[1] 0.9716
```

As seen here, the Random Forest model provides better accuracy (96%) over the K-Nearest Neighbor model (93%). Sensitivity and specificity also look very good. Sensitivity is at 94% and specificity at 97%. The Random Forest model is a great model for predicting customer satisfaction.

We can also see a hierarchical list of the importance of the variables. The top 5 variables are:

```
varImp(train_rf)
```

```
rf variable importance

   Importance
9      100.00
4       79.35
6       76.56
20      70.44
14      69.55
15      52.05
21      43.94
22      42.06
19      42.00
16      36.85
7       33.39
17      33.16
18      29.09
11      24.04
13      21.90
10      19.51
12      17.95
5        8.92
8        4.15
3        0.00
```

09 Inflight wifi Service (Satisfaction Level)
04 Customer Type (Loyal or Disloyal
06 Age
20 Check In Service (Satisfaction Level)
14 Online Boarding (Satisfaction Level)

Can we do even better by combining models? Let's take a look at an ensemble model.

Ensemble Model

The idea of an ensemble is similar to the idea of combining data from different pollsters to obtain a better estimate of the true support for each candidate. In machine learning, we can usually greatly improve the final results by combining the results of different algorithms.

My Ensemble model is based on the predictions from the top two models created previously. These models are the K-Nearest Neighbor and Random Forest models.

```
ensemble <- cbind(knn = knn_preds == 0,
rf = rf_preds == 0)
ensemble_preds <- ifelse(rowMeans(ensemble) > 0.5, 0, 1)
mean(ensemble_preds == test_y)
```

```
[1] 0.9516
```

```
sensitivity(factor(ensemble_preds), test_y, positive = 1) # satisfied customer
```

```
[1] 0.9496
```

```
specificity(factor(ensemble_preds), test_y, negative = 0) # not satisfied or neutral customer
```

```
[1] 0.9531
```

The Ensemble model gives us close to the same accuracy as the Random Forest model (95% versus 96%). However, the Ensemble model gives us slightly more sensitivity (95% versus 94%) at the expense of specificity (95% versus 97%). So, which model you consider better, Ensemble or Random Forest, will depend whether you want better sensitivity (ability to predict satisfied customers) or better specificity (ability to predict not satisfied or neutral customers).

Accuracy Comparison Table

Here is a table displaying my modeling results:

```
models <- c("K means", "K nearest neighbors", "Random forest", "Ensemble")
accuracy <- c(mean(kmeans_preds == test_y),
mean(knn_preds == test_y),
mean(rf_preds == test_y),
mean(ensemble_preds == test_y))
sensitivity <- c(sensitivity(factor(kmeans_preds), test_y, positive = 1),
sensitivity(factor(knn_preds), test_y, positive = 1),
sensitivity(factor(rf_preds), test_y, positive = 1),
sensitivity(factor(ensemble_preds), test_y, positive = 1))
specificity <- c(specificity(factor(kmeans_preds), test_y, negative = 0),
specificity(factor(knn_preds), test_y, negative = 0),
specificity(factor(rf_preds), test_y, negative = 0),
specificity(factor(ensemble_preds), test_y, negative = 0))
data.frame(Model=models, Accuracy=accuracy, Sensitivity=sensitivity, Specificity=specificity)
```

```
                Model Accuracy Sensitivity Specificity
1             K means   0.7894      0.8614      0.7350
2 K nearest neighbors   0.9337      0.8953      0.9628
3       Random forest   0.9583      0.9408      0.9716
4            Ensemble   0.9516      0.9496      0.9531
```

In summary, although the Random Forest model has the greatest accuracy, the ensemble model gives us more ability to correctly predict satisfied airline passengers. If we are focused on what makes our customers happy then the ensemble model is best. However, if we are focused on what makes the customer unhappy (so that we can change those factors), then the Random Forest model is our best bet.

3. Results

For this project, I applied machine learning techniques to a dataset found on the Kaggle website called Airline Passenger Satisfaction. This dataset contains the details of an airline passenger satisfaction survey.

My goal was to explore the data to determine its usability and then apply various machine learning algorithms in the caret package to determine which one most correctly identifies the satisfied airline passenger.

I explored the data to determine whether or not I could use various regression models based on their ability to combine variables and improve the modeling process. I started my exploration with hierarchical clustering and then applied Principal Components Analysis. Based on these observations, I concluded that the dataset is amenable to applying machine learning techniques that are based on their ability to combine variables using machine learning techniques such as clustering, nearest neighbor, and random forest.

Then, using the caret package, I applied three machine learning models: K-Means Clustering, K-Nearest Neighbors, and Random Forest. I also created an Ensemble model using the top two best machine learning models. The top two models are the K-Nearest Neighbors and the Random Forest models. As a result of my analyses, I determined that application of the Random Forest model provides the most accuracy; however, the Ensemble model, close behind in accuracy, actually provides better sensitivity, or a better ability to correctly predict satisfied airline passengers. Here is a summary of my results:

Model	Accuracy	Sensitivity	Specificity
K Means	0.7894	0.8614	0.7350
K Nearest Neighbors	0.9337	0.8953	0.9628
Random Forest	0.9583	0.9408	0.9716
Ensemble	0.9516	0.9496	0.9531

4. Conclusion

Based on my exploration of the data I determined that I could use four specific machine learning models to analyze the Airline Passenger Satisfaction dataset. Using the caret package, I applied three machine learning models: K-Means Clustering, K-Nearest Neighbors, and Random Forest, and then, created an Ensemble model using the top two best machine learning models. The top two models are the K-Nearest Neighbors and the Random Forest models. As a result of these analyses, I determined that application of the Random Forest model provides the most accuracy; however, the Ensemble model, close behind in accuracy, actually provides better sensitivity, or a better ability to correctly predict satisfied airline passengers.

By way of further analysis, I believe that the survey needs improvement by combining like variables into fewer survey questions and by making some variables more direct. For example, we saw that questions about wifi and online activities formed a single cluster. So, maybe we could combine these questions into one or two questions that incorporate all of the web-based questions. Also, we could word the variables that relate to flying distances and arrival/departure delays as more direct questions that ask for customer satisfaction regarding those items. In the process of taking these steps, I believe that the surveyors may actually gain further insights into what makes the airline passenger happy.

Also, my application of advanced machine learning was not exhaustive. We should look at other possible models for analyzing the dataset.

This survey has a lot of potential for improving airline passenger satisfaction, but the survey itself needs improvement.