Customer RFM Analysis, Segmentation and Attrition Prediction

Madan Kundapur

08/01/2020

Please find project files in GitHub at https://github.com/madankundapur/ProjectX.git

1. Introduction:

One of the most important tasks for any business is to know their customers. In today's world every business needs to offer personalized products and services to its customers or risk losing them.

Customers are both similar and different. It is impossible to have individualized products and services for each customer. Hence the need to segment customers with similar characteristics and have tailored offerings to each group.

There are many characteristics on which customers can be segmented. Common characteristics used are customer behaviour, demography and interests.

Data like customer purchase date and value are readily available with vendors. It therefore makes sense to use them for targeted marketing. Recency of purchase, Frequency of purchases and Monetary value of purchases - popularly referred to as RFM (Receny-Frequency-Monetary) are one of the most effective methods used for customer segmentation.

RFM analysis is also a good customer attrition indicator because it examines how recently a customer has purchased, how often they purchase and how much they usually spend. It is easy to detect if there's a drop in customer's purchases or average spend and identify customers who may attrite.

1.1 Objective:

The purpose of this project is to use Recency, Frequency and Monetary value to group customers into High, Medium and Low segments by applying a clustering algorithm and then predict customer attrition.

We will follow these steps in our analysis:

- Pick a dataset and tidy it up for analysis.
- Do an initial RFM analysis using simple scoring.
- Use clustering algorithms to segment customers.
- Use an ensemble of models to predict customer attrition.
- Analyse the results of RFM analysis, predictions and share insights.

A bit about clustering and k-means

Clustering algorithms do not necessarily know the outcomes and are interested in discovering groups. K-means is one of the most popular ways to segment customers using unsupervised clustering techniques.

Given a set of observations (x1, x2, …, xn), where each observation is a d-dimensional real vector, k-means clustering aims to partition n observations into k (≤ n) sets S = {S1, S2, …, Sk} so as to minimize the within-cluster sum of squares… more about k-means

We will use k-means for customer segmentation.

1.2 Data:

The dataset of Superstore Orders is sourced from Tableau Community forum and is representative. Ideally, a larger dataset of real orders would have yielded better insights. Nevertheless, it works well enough to illustrate the objectives defined for this project.

Load dependent libraries

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

Load data from CSV file

```
# Load store orders data from csv file
orders <- read_csv("SuperstoreOrders.csv")
```

The dataset has 9994 observations with 21 variables and contains store orders data for the United States.

```
str(orders)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 9994 obs. of  21 variables:
##  $ Row ID       : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Order ID     : chr  "CA-2017-152156" "CA-2017-152156" "CA-2017-138688" "US-2016-108966" ...
##  $ Order Date   : chr  "11/8/2017" "11/8/2017" "6/12/2017" "10/11/2016" ...
##  $ Ship Date    : chr  "11/11/2017" "11/11/2017" "6/16/2017" "10/18/2016" ...
##  $ Ship Mode    : chr  "Second Class" "Second Class" "Second Class" "Standard Class" ...
##  $ Customer ID  : chr  "CG-12520" "CG-12520" "DV-13045" "SO-20335" ...
##  $ Customer Name: chr  "Claire Gute" "Claire Gute" "Darrin Van Huff" "Sean O'Donnell" ...
##  $ Segment      : chr  "Consumer" "Consumer" "Corporate" "Consumer" ...
##  $ Country      : chr  "United States" "United States" "United States" "United States" ...
##  $ City         : chr  "Henderson" "Henderson" "Los Angeles" "Fort Lauderdale" ...
##  $ State        : chr  "Kentucky" "Kentucky" "California" "Florida" ...
##  $ Postal Code  : num  42420 42420 90036 33311 33311 ...
##  $ Region       : chr  "South" "South" "West" "South" ...
##  $ Product ID   : chr  "FUR-BO-10001798" "FUR-CH-10000454" "OFF-LA-10000240" "FUR-TA-10000577"
...
##  $ Category     : chr  "Furniture" "Furniture" "Office Supplies" "Furniture" ...
##  $ Sub-Category : chr  "Bookcases" "Chairs" "Labels" "Tables" ...
##  $ Product Name : chr  "Bush Somerset Collection Bookcase" "Hon Deluxe Fabric Upholstered
Stacking Chairs, Rounded Back" "Self-Adhesive Address Labels for Typewriters by Universal"
"Bretford CR4500 Series Slim Rectangular Table" ...
##  $ Sales        : num  262 731.9 14.6 957.6 22.4 ...
##  $ Quantity     : num  2 3 2 5 2 7 4 6 3 5 ...
##  $ Discount     : num  0 0 0 0.45 0.2 0 0 0.2 0.2 0 ...
##  $ Profit       : num  41.91 219.58 6.87 -383.03 2.52 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   `Row ID` = col_double(),
##   ..   `Order ID` = col_character(),
##   ..   `Order Date` = col_character(),
##   ..   `Ship Date` = col_character(),
##   ..   `Ship Mode` = col_character(),
```

```
##   ..   `Customer ID` = col_character(),
##   ..   `Customer Name` = col_character(),
##   ..   Segment = col_character(),
##   ..   Country = col_character(),
##   ..   City = col_character(),
##   ..   State = col_character(),
##   ..   `Postal Code` = col_double(),
##   ..   Region = col_character(),
##   ..   `Product ID` = col_character(),
##   ..   Category = col_character(),
##   ..   `Sub-Category` = col_character(),
##   ..   `Product Name` = col_character(),
##   ..   Sales = col_double(),
##   ..   Quantity = col_double(),
##   ..   Discount = col_double(),
##   ..   Profit = col_double()
##   .. )
```

Data variable names have spaces in them and as a practise it is good to avoid spaces. Also note that the variable names are in proper casing - we will retain and follow that convention for naming data variables.

```
names(orders)<-str_replace_all(names(orders), c(" " = ""))
```

OrderDate variable is of type 'character'. Changing it to 'date'

```
orders$OrderDate <- as.Date(orders$OrderDate, "%m/%d/%Y")
class(orders$OrderDate)
```

```
## [1] "Date"
```

To keep data tidy check for duplicates and filter them out.

```
duplicates <- which(duplicated(orders))
duplicates
```

```
## integer(0)
```

```
# No duplicates exist in data
rm(duplicates)
```

Data that we need for RFM analysis is *OrderDate* and *Sales* amount by customer. The dataset has order details at the product level which we don't need. Let us aggregate *Sales* amount and select only necessary variables for further analysis

```
orders <- orders %>%
  group_by(CustomerID, OrderID , OrderDate) %>%
  summarize(Sales = sum(Sales)) %>%
  select(CustomerID, OrderID , OrderDate, Sales)

# Checking if the orders are equal to the observations in the dataset
length(unique(orders$OrderID ))
```

```
## [1] 5009
```

```
nrow(orders)
```

```
## [1] 5009
```

Order dates range from Jan 2015 through Dec 2018. Compute maximum date from the dataset. This will help compute *DaysSincePurchase* and *Recency*. Note that a day is added to the maximum date to ensure that there are no zeroes calculated (applies to purchases made on the last day).

```
range(orders$OrderDate)
```

```
## [1] "2015-01-03" "2018-12-30"
```

```
max_date <- max(orders$OrderDate)+1
```

Compute *PurchaseYear* - which is year part of order date and *DaysSincePurchase* - which is the difference between order date and the maximum date in the dataset

```
orders <- orders %>%
  mutate(PurchaseYear = as.numeric(format(OrderDate, "%Y")),
         DaysSincePurchase = as.numeric(difftime(max_date, OrderDate,"days")))

rm(max_date)
```

1.3 Compute Recency, Frequency and Monetary Value:

For each customer compute RFM values:

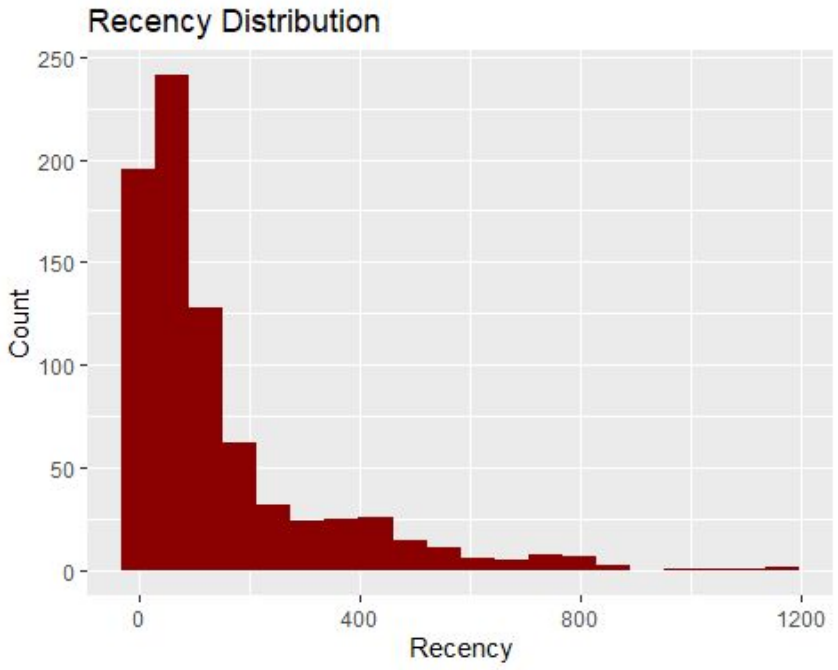
- *Recency* is the duration in days since the last purchase made by the customer
- *Frequency* is the number of distinct orders by customer
- *Monetary* value is total sales amount for the customer

```
customers <- orders %>%
  group_by(CustomerID) %>%
  summarise(Recency = min(DaysSincePurchase),
            Frequency = n_distinct(OrderID),
            Monetary = sum(Sales))

knitr::kable(summary(customers))
```

CustomerID	Recency	Frequency	Monetary
Length:793	Min. : 1.0	Min. : 1.000	Min. : 4.833
Class :character	1st Qu.: 31.0	1st Qu.: 5.000	1st Qu.: 1146.050
Mode :character	Median : 76.0	Median : 6.000	Median : 2256.394
NA	Mean : 147.8	Mean : 6.317	Mean : 2896.848
NA	3rd Qu.: 184.0	3rd Qu.: 8.000	3rd Qu.: 3785.276
NA	Max. :1166.0	Max. :17.000	Max. :25043.050

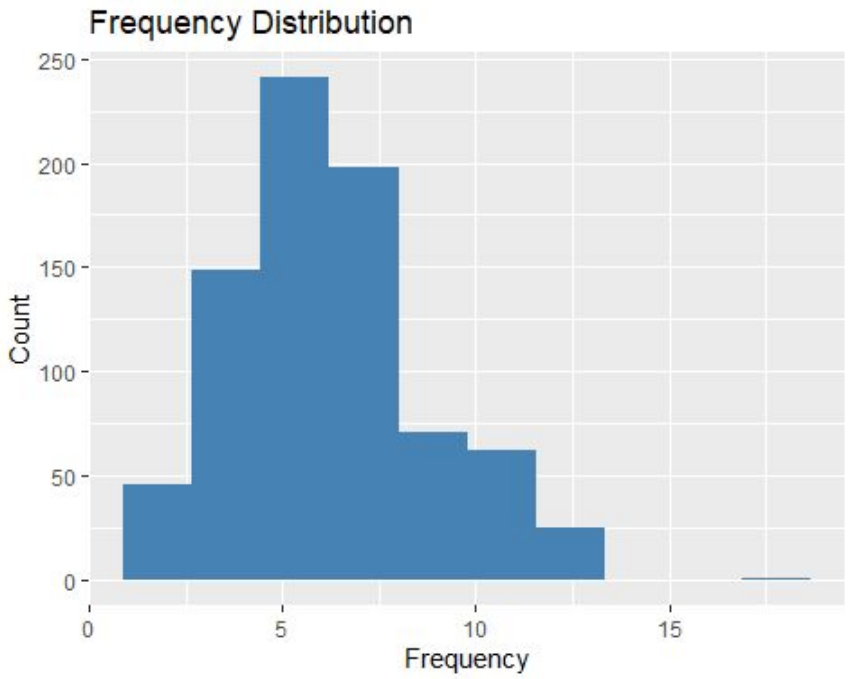
Plot distribution for Recency, Frequency and Monetary Value to explore RFM data

```
customers %>% ggplot(aes(Recency))  +
  geom_histogram(bins=20,fill = "darkred") +
  labs(x = "Recency", y = "Count", title = "Recency Distribution")
```



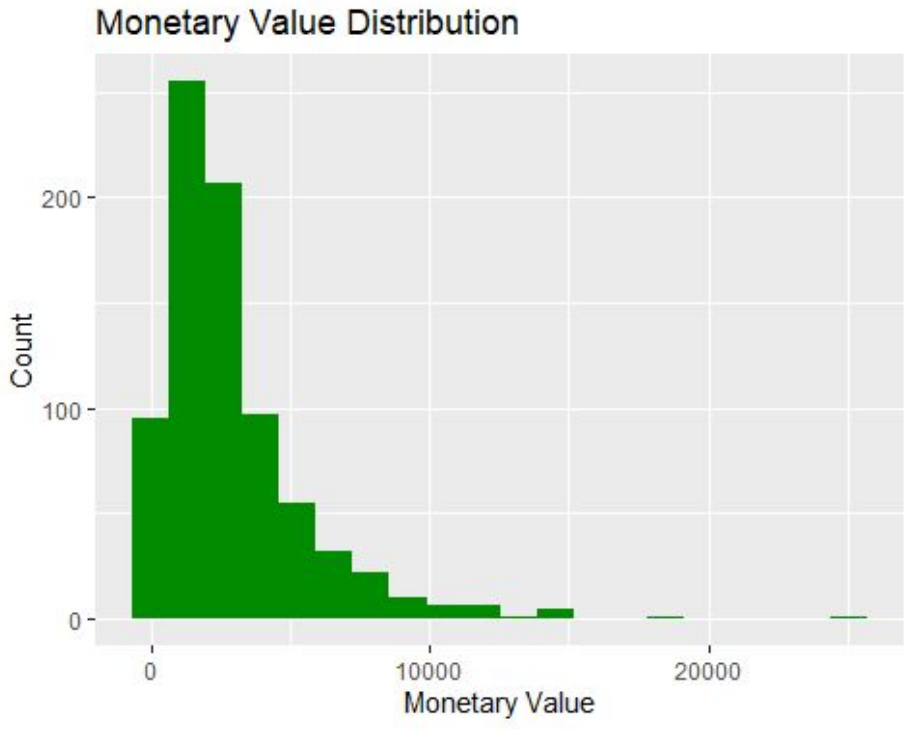
From the Recency plot, more than 80% of customers have been active in the last one year, which is a good sign.

```
customers %>% ggplot(aes(Frequency))  +
  geom_histogram(bins=10,fill = "steelblue")+
  labs(x = "Frequency", y = "Count", title = "Frequency Distribution")
```



From the Frequency plot, the values are more-or-less distributed and the range is between 1 and 13 with an outlier of 17.

```
customers %>% ggplot(aes(Monetary))  +
  geom_histogram(bins=20,fill = "green4") +
  labs(x = "Monetary Value", y = "Count", title = "Monetary Value Distribution")
```



From the Monetary value plot, more than 97% of customers have spent less than $10000 across years.

Since the scale of values are very different for Recency, Frequency and Monetary. Let us remove the skew and standardise the RFM values.

```
customers$RecencyZ <- scale(log(customers$Recency), center=TRUE, scale=TRUE)
customers$FrequencyZ <- scale(log(customers$Frequency), center=TRUE, scale=TRUE)
customers$MonetaryZ <- scale(log(customers$Monetary), center=TRUE, scale=TRUE)
```

We now have a tidy dataset with 793 observations of 8 variables to work with.

---

2. Analysis:

2.1 Simple RFM Scoring:

Let us first do some initial analysis and assign scores to RFM values. Using *ntile* analytic function divide Recency, Frequency and Monetary value into 3 buckets.

A value of 1 means Low, 2 means Medium and 3 means High.

Recency score *RScore* is based on the recency of purchase. Lower the Recency, the better the customer - note the code uses *desc* for the *ntile* function.

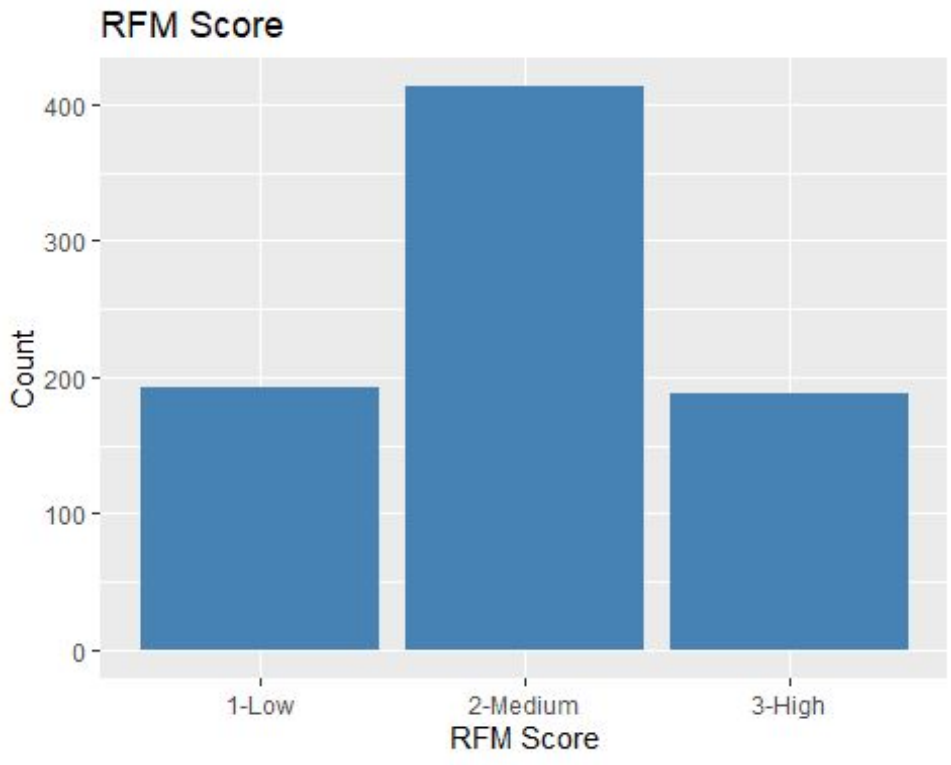
Frequency score *FScore* is based on the number of orders made by the customer. Higher the frequency, the better the customer.

Monetary value score *MScore* is based on the total value of sales by customer. Higher the Recency, the better the customer.

RFMScore is the mean of Recency, Frequency & Monetary Scores

RFMScoreLabel is a label assigned based on *RFMScore*

```
customers <- customers %>%
  mutate(RScore = ntile(desc(Recency),3),
         FScore = ntile(Frequency,3),
         MScore = ntile(Monetary,3),
         RFMScore = round((RScore+FScore+MScore)/3,0),
         RFMScoreLabel = case_when(RFMScore == 1 ~ "1-Low", RFMScore == 2 ~ "2-Medium", RFMScore
== 3 ~ "3-High"))

table(customers$RFMScore)

##
##   1   2   3
## 192 413 188

customers %>% ggplot(aes(RFMScoreLabel)) +
  geom_bar(fill = "steelblue") +
  labs(x = "RFM Score", y = "Count", title = "RFM Score")
```



The RFM Score plot shows that more than 50% of customers are in the Medium value range. High and Low value customers are almost equally distributed. Let us check later in the analysis and see if High value customers are at risk.

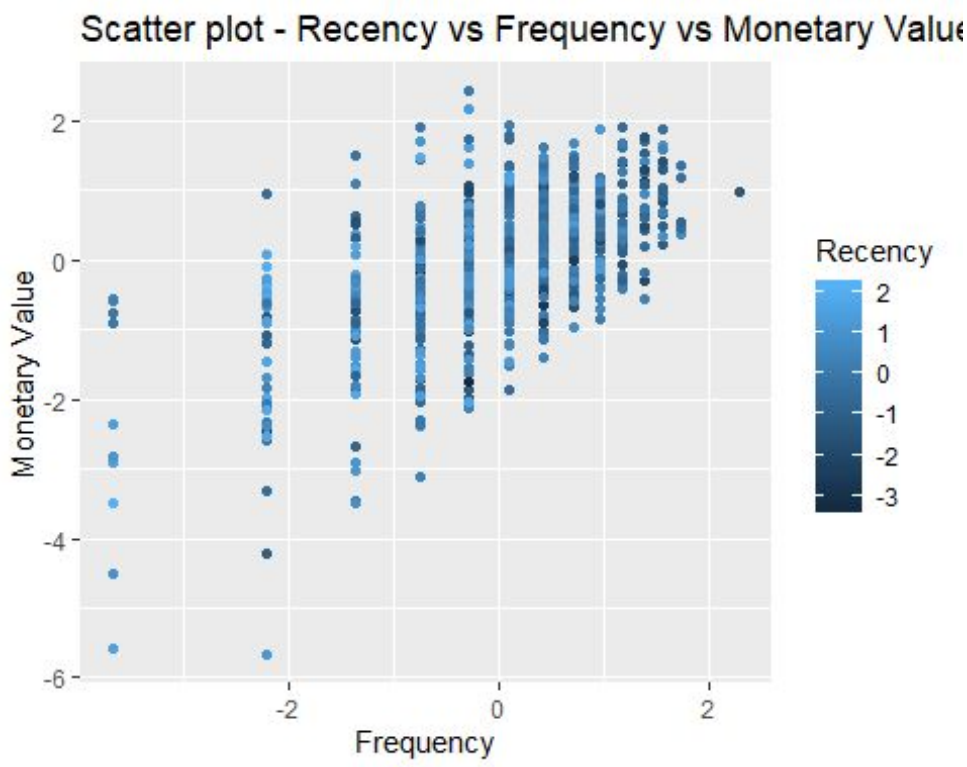
2.2 Clustering:

As mentioned earlier, for the purpose of this analysis we will use k-means clustering algorithm.

To use k-means clustering algorithm we have to pre-define k, which is the number of clusters we want to define. The k-means algorithm is iterative. The first step is to define k centers. Then each observation is assigned to the cluster with the closest center to that observation. In the second step the centers are redefined using the observation in each cluster. The column means are used to define a centroid. We repeat these steps until the centers converge.

Let us first see how the scatter plot looks for RFM values in the final dataset

```
customers %>% ggplot(aes(x=FrequencyZ,y=MonetaryZ)) +
  geom_point(aes(colour = RecencyZ)) +
  scale_colour_gradient(name="Recency") +
  labs(x = "Frequency", y = "Monetary Value", title = "Scatter plot - Recency vs Frequency vs
Monetary Value")
```



Note that the plot shows high-frequency and high monetary value customers in the top right with recency indicated in dark shades of blue. Similarly, low-frequency, low monetary value customers are in the bottom-left with recency indicated in the lighter shades of blue.

Looking at the entire plot it is hard find clusters. But the data points also don't seem to be distributed continuously. We therefore need to assume the clusters to extract and see which is the best fit. Let us assume a maximum value of 10 for cluster centers and loop through to arrive at the best cluster.

We will print the medians of RFM grouped by the cluster levels to analyze and find the best cluster.

Each cluster also provides us with the following information.

- *totss* is the total sum of squares
- *withinss* is the vector of within-cluster sum of squares, one component per cluster

- *tot.withinss* is the total within-cluster sum of squares, i.e. sum(withinss)
- *betweenss* is the between-cluster sum of squares, i.e. totss-tot.withinss

Also, let us temporarily persist *tot.withinss* and plot it to identify the best cluster.

```
j<- 10

# data frame to hold cluster components
ss <- data.frame(K=integer(),
                 TWSS=numeric())

# ensure customers dataset is a data frame
customers <- as.data.frame(customers)

# loop to create upto 10 clusters
for (i in 1:j ) {

  set.seed(1, sample.kind="Rounding")

  # Run k-means with i centers, assume nstart =25
  km <- kmeans(customers[,c(5:7)], centers = i, nstart = 25)

  # Adding cluster data  to customers dataset for each i in different variables
  col_nm <- paste("C", i, sep="")
  customers[,(col_nm)] <- factor(km$cluster, levels = c(1:i))

  # Find medians for RFM grouped by cluster and print them
  med <- customers %>%
      group_by(Cluster = customers[,(col_nm)]) %>%
      summarize(Recency=round(median(Recency), 0),
              Frequency=round(median(Frequency),1),
              Monetary=round(median(Monetary),2))
  print(paste(i, "Clusters", sep=" "))
  print(med)

  # store cluster info
  ss[i,("K")] <- i
  ss[i,("TWSS")] <- km$tot.withinss
}
```

```
## [1] "1 Clusters"
## # A tibble: 1 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1            76         6    2256.
## [1] "2 Clusters"
## # A tibble: 2 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1            49         7    3087.
## 2 2           208         4     935.
## [1] "3 Clusters"
## # A tibble: 3 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1            27         8    3101.
## 2 2           120         6    2459.
```

```
## 3 3           204         3     574.
## [1] "4 Clusters"
## # A tibble: 4 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1            65         8    4047.
## 2 2            16         7    2217 
## 3 3           160         5    1553.
## 4 4           284         2     348.
## [1] "5 Clusters"
## # A tibble: 5 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1           296         2     219.
## 2 2           303         5    1418.
## 3 3            44         5     904.
## 4 4            98         7    3492.
## 5 5            22         8    3101.
## [1] "6 Clusters"
## # A tibble: 6 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1            56         5     913.
## 2 2            38         8    3336.
## 3 3           153         7    2965.
## 4 4             7         7    2550.
## 5 5           407         4    1057.
## 6 6           227         2     131.
## [1] "7 Clusters"
## # A tibble: 7 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1           246         2     84.0
## 2 2            42         4    862.
## 3 3           175         5   1791.
## 4 4           453         3    886.
## 5 5            34         8   2929.
## 6 6           108         8   5013.
## 7 7             7         7   2562.
## [1] "8 Clusters"
## # A tibble: 8 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1            35         9   3013.
## 2 2           265         2     79.8
## 3 3            40         4    741.
## 4 4           136         8   4873.
## 5 5           203         5   1523.
## 6 6             7         7   2528.
## 7 7           445         3    893.
## 8 8            51         5   3079.
## [1] "9 Clusters"
## # A tibble: 9 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1           378         5   2558.
## 2 2           126         6   1082.
## 3 3            99         8   4642.
## 4 4            31         4    742.
```

```
## 5 5           430         3    863.
## 6 6            55         5   2932.
## 7 7             6         7   2493.
## 8 8            29         8   3037.
## 9 9           265         2     79.8
## [1] "10 Clusters"
## # A tibble: 10 x 4
##    Cluster Recency Frequency Monetary
##    <fct>     <dbl>     <dbl>    <dbl>
##  1 1            31         4    740.
##  2 2           419         3    917.
##  3 3            30         9   4493.
##  4 4           192         5    839.
##  5 5            59         5   3021.
##  6 6           346         5   2367.
##  7 7           265         2     79.8
##  8 8             7         7   2493.
##  9 9           143         8   4511.
## 10 10           56         7   1860.
```

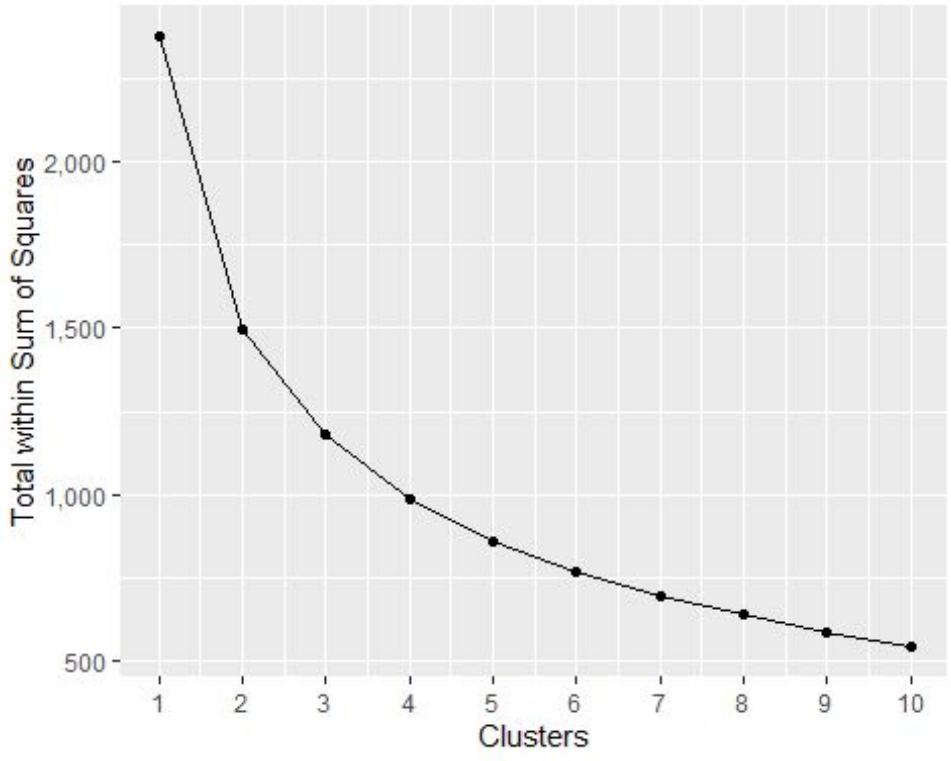
In the 2-cluster resultset, we find that high-recency, high-frequency and high-value customers are in one cluster. Low-recency, low-frequency and low-value customers are in the other cluster.

In the 3-cluster resultset, we find that high-recency, high-frequency and high-value customers are in cluster 1. Medium-recency, medium-frequency and medium-value customers are in cluster 2. Low-recency, low-frequency and low-value customers are in cluster 3.

In the 4-cluster resultset, cluster 1 has high-recency, high-frequency and high-value customers; cluster 2 has high-recency, high-frequency and medium-value customers; cluster 3 has medium-recency, medium-frequency and medium-value customers; cluster 4 has low-recency, low-frequency and low-value customers.

3-cluster resultset seems interpretable. Let us plot 'Total within Sum of Squares' against k to see if we can find an elbow at cluster 3. An 'elbow' indicates the most optimal k.

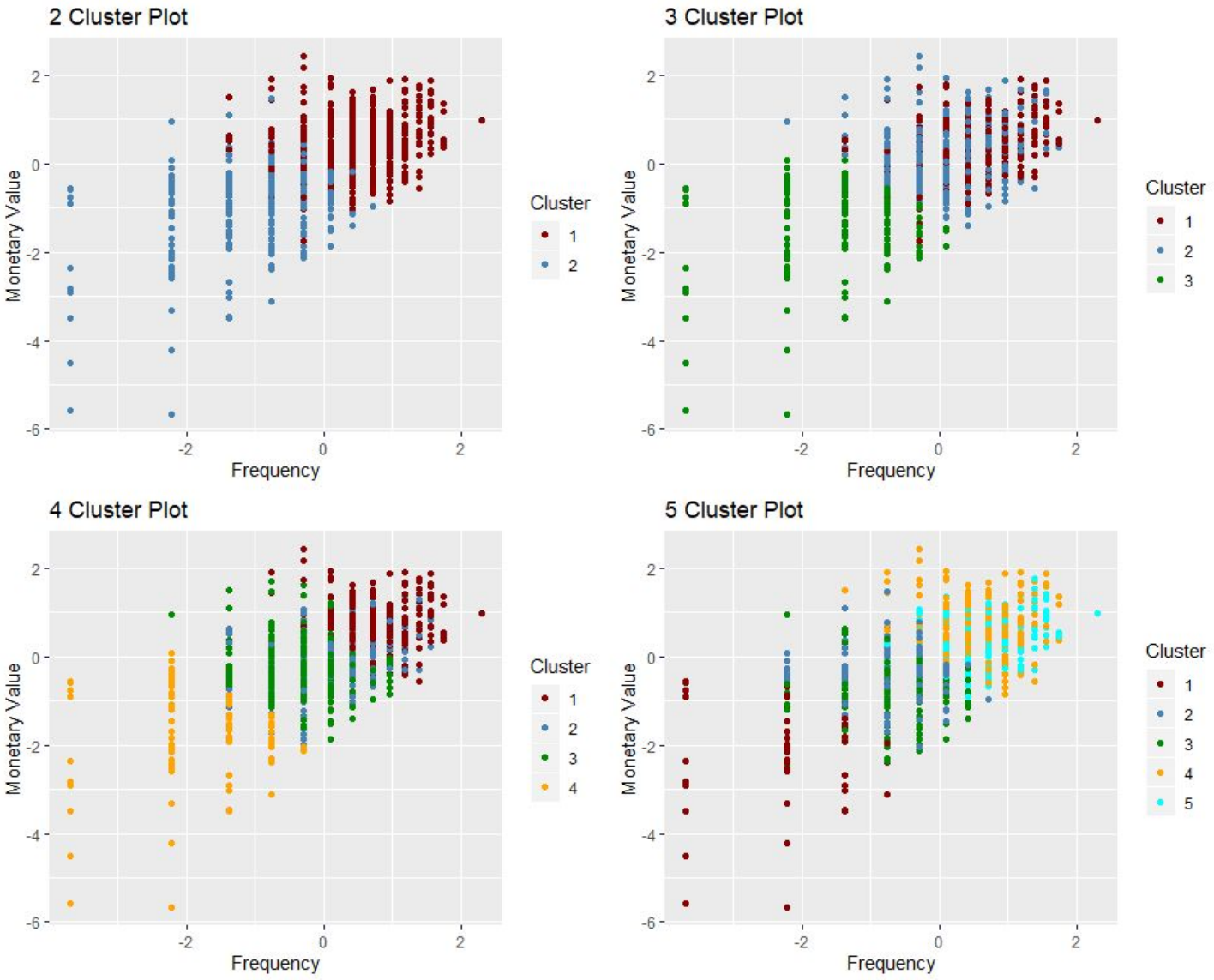
```r
# Plot sum within sum of squares
ss %>% ggplot(aes(x = K, y = TWSS)) +
  geom_point() +
  geom_line()+
  scale_y_continuous(labels = scales::comma)+
  scale_x_continuous(breaks = 1:j)+
  xlab("Clusters")+
  ylab("Total within Sum of Squares")
```



We do find the bend at cluster 3. or is it at cluster 2? Let us plot cluster solutions from 2 to 5 and see how they look visually.

```
# color palette for the scatter plot
palette <- c('darkred','steelblue','green4','orange', "cyan")

# Plot RFM flor cluster 2
p1 <- customers %>% ggplot( aes(x = FrequencyZ, y = MonetaryZ))+
  geom_point(aes(colour = C2))+
  scale_colour_manual(name = "Cluster", values=palette)+
  xlab("Frequency")+
  ylab("Monetary Value")+
  ggtitle(paste("2 Cluster Plot", sep=" "))

# Plot RFM flor cluster 3
p2<- customers %>% ggplot( aes(x = FrequencyZ, y = MonetaryZ))+
  geom_point(aes(colour = C3))+
  scale_colour_manual(name = "Cluster", values=palette)+
  xlab("Frequency")+
  ylab("Monetary Value")+
  ggtitle(paste("3 Cluster Plot", sep=" "))

# Plot RFM flor cluster 4
p3<- customers %>% ggplot( aes(x = FrequencyZ, y = MonetaryZ))+
  geom_point(aes(colour = C4))+
  scale_colour_manual(name = "Cluster", values=palette)+
  xlab("Frequency")+
  ylab("Monetary Value")+
  ggtitle(paste("4 Cluster Plot", sep=" "))

# Plot RFM flor cluster 5
p4<- customers %>% ggplot( aes(x = FrequencyZ, y = MonetaryZ))+
  geom_point(aes(colour = C5))+
```

```
  scale_colour_manual(name = "Cluster", values=palette)+
  xlab("Frequency")+
  ylab("Monetary Value")+
  ggtitle(paste("5 Cluster Plot", sep=" "))

# Arrange plots
grid.arrange(p1, p2, p3, p4, ncol=2, nrow = 2)
```



Cluster 2 may be simplistic. Cluster 3 does look good. It may not always be possible to find distinct clusters since it depends on the data. For the purpose of this project though the idea is to find High value customers and see if they are retained.

```
rm(ss, med, col_nm, i, j, km, palette, p1, p2, p3, p4)
```

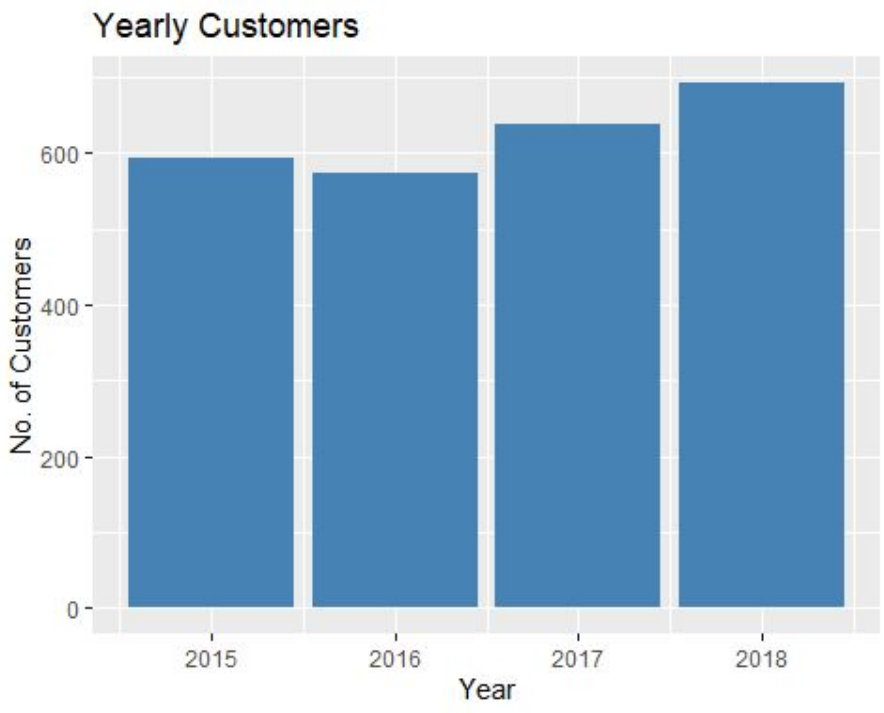
2.3 Yearly Orders Analysis

Let us now look at how the orders are stacked up by year.

```
# Summarize total customers, orders & sales by year
yearly_orders <- orders %>%
  group_by(PurchaseYear) %>%
  summarise(Customers = n_distinct(CustomerID),
            Orders = n_distinct(OrderID),
            Sales = sum(Sales))

yearly_orders
```

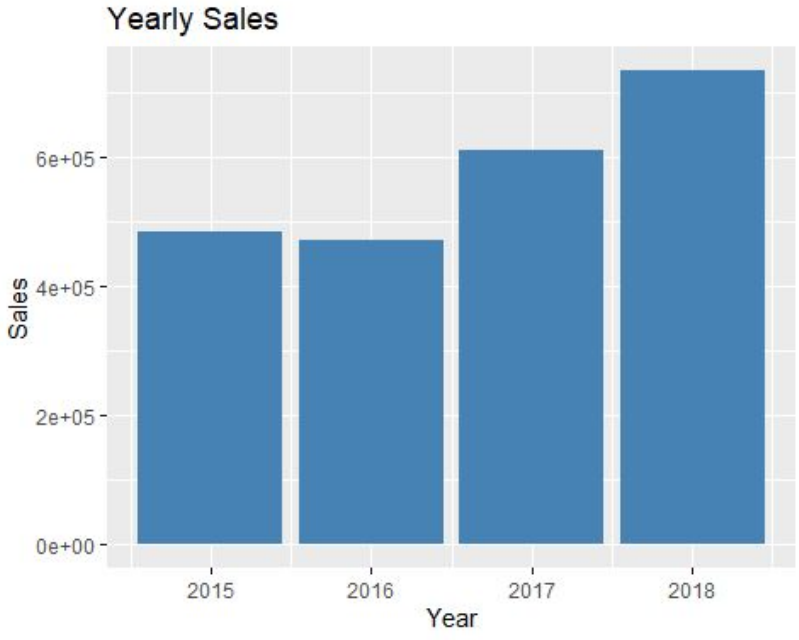
```
## # A tibble: 4 x 4
##   PurchaseYear Customers Orders   Sales
##          <dbl>     <int>  <int>   <dbl>
## 1         2015       595    969 484247.
## 2         2016       573   1038 470533.
## 3         2017       638   1315 609206.
## 4         2018       693   1687 733215.
```

```
# Plot Number of Customers by year
yearly_orders %>% ggplot(aes(x = PurchaseYear, y = Customers)) +
  geom_bar(stat = "identity", fill = 'steelblue') +
  labs(x = "Year", y = "No. of Customers", title = "Yearly Customers")
```



```r
# Plot Orders by year
yearly_orders %>% ggplot(aes(x = PurchaseYear, y = Orders)) +
  geom_bar(stat = "identity", fill = 'steelblue') +
  labs(x = "Year", y = "Orders", title = "Yearly Orders")
```


```r
# Plot Total Sales by year
yearly_orders %>% ggplot(aes(x = PurchaseYear, y = Sales)) +
  geom_bar(stat = "identity", fill = 'steelblue') +
  labs(x = "Year", y = "Sales", title = "Yearly Sales")
```




```r
rm(yearly_orders,p1,p2,p3)
```

Notice that there is a dip in the number of customers and sales in 2016. However, the numbers of orders have increased. Otherwise, there has been a gradual increase in customers, orders and sales.

2.4 Preparing data for attrition prediction

First, let us group customer orders by year and arrange it in the form of a cross-tab

```
# Compute orders by year and arrange in a cross tab
customer_orders_by_year <- orders %>%
       group_by(CustomerID, PurchaseYear) %>%
       summarize(count = n()) %>%
       spread(PurchaseYear, count)

head(customer_orders_by_year)
```

```
## # A tibble: 6 x 5
## # Groups:   CustomerID [6]
##   CustomerID `2015` `2016` `2017` `2018`
##   <chr>       <int>  <int>  <int>  <int>
## 1 AA-10315        2      1      1      1
## 2 AA-10375        2      3      2      2
## 3 AA-10480        1     NA      2      1
## 4 AA-10645        2      1      2      1
## 5 AB-10015        2     NA      1     NA
## 6 AB-10060       NA      1      3      4
```

Notice that the dataset has NA's. Replace NA's with zero.

```
# updates NA's with 0
customer_orders_by_year[is.na(customer_orders_by_year)] = 0
```

Column names of years are numeric. Change it to apha-numeric by prefixing a 'Y'

```
# change names of years (numbers) to Y+ year
names(customer_orders_by_year)<-str_replace_all(names(customer_orders_by_year), c("2" = "Y2"))
```

Now to arrive at customer status, let us use the count of orders. if there are no orders in the later years, it means customer has 'Attrited'. If not, customer is 'Retained'. Note that there are customers who may not have any orders in between, but have come back. In such cases we mark them as 'Retained'

```
# Update customer status -  Attrited or Retained
customer_orders_by_year<- customer_orders_by_year %>%
                          mutate(Status = case_when(Y2018 == 0 ~ "Attrited",
                                                    Y2018 == 0 &  Y2017 == 0 ~ "Attrited",
                                                    Y2018 == 0 &  Y2017 == 0 & Y2016 == 0 ~
"Attrited",
                                    TRUE ~ "Retained"))
```

Merge *customer_orders_by_year* dataset with *customers* dataset for further analysis

```
# Merge Customer Status with customers dataset
customers <- merge(customers, customer_orders_by_year,  by="CustomerID", all=TRUE, sort=TRUE)

rm(customer_orders_by_year)
```

Now, let us pick the required variables for prediction.

We want to predict Status using predictors - Recency, Frequency & Monetary value.

```
# Select required variables for predicting status
# select data for last year and prior for training and testing
customer_status <- customers %>%
  select(CustomerID, Recency, Frequency, Monetary, Status)

# Update Status class to factor
customer_status$Status <- factor(customer_status$Status)

# Update customer id as row names and remove customer id variable
row.names(customer_status) <- customer_status$CustomerID
customer_status <- select(customer_status, -CustomerID)

# Status variable distribution
table(customer_status$Status)

##
## Attrited Retained
##      100      693
```

2.5 Using models for prediction

Let us now partition data into a training set and test set. We will then apply different algorithms to arrive at the best fit based on accuracy.

We will use Logistic regression, Linear discriminant analysis, Quadratic discriminant analysis, K nearest neighbors and Random Forest algorithms. We will also create an Ensemble and then find the model that gives the best accuracy.

For each model we will first tune the model by using the *train* function. We will then use the *predict* function to apply the model on the test set. For each model we will then compute accuracy.

```
# Create training data and test data
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(customer_status$Status, times = 1, p = 0.2, list = FALSE)
train_set <- customer_status[-test_index,]
test_set <- customer_status[test_index,]

# Generalized linear model
train_glm <- train(Status ~ ., data = train_set, method = "glm")
glm_preds <- predict(train_glm, test_set)
mean(glm_preds == test_set$Status)

## [1] 0.9874214

# Linear discriminant analysis
train_lda <- train(Status ~ ., data = train_set, method = "lda")
lda_preds <- predict(train_lda, test_set)
mean(lda_preds == test_set$Status)

## [1] 0.9874214

# Quadratic discriminant analysis
train_qda <- train(Status ~ ., data = train_set, method = "qda")
```

```r
qda_preds <- predict(train_qda, test_set)
mean(qda_preds == test_set$Status)
```

```
## [1] 0.9685535
```

```r
# K Nearest Neighbour
train_knn <- train(Status ~ ., data = train_set, method = "knn", tuneLength=20)
knn_preds <- predict(train_knn, test_set)
mean(knn_preds == test_set$Status)
```

```
## [1] 0.9622642
```

```r
# Random Forest
train_rf <- train(Status ~ ., data = train_set, method = "rf", tuneLength=20, importance = TRUE)
```

```
## note: only 2 unique complexity parameters in default grid. Truncating the grid to 2 .
```

```r
rf_preds <- predict(train_rf, test_set)
mean(rf_preds == test_set$Status)
```

```
## [1] 0.9937107
```

```r
# Ensemble
ensemble <- cbind(glm = glm_preds == test_set$Status,
                  lda = lda_preds == test_set$Status,
                  qda = qda_preds == test_set$Status,
                  knn = knn_preds == test_set$Status,
                  rf = rf_preds == test_set$Status)
ensemble_preds <- ifelse(rowMeans(ensemble) > 0.5, "Retained", "Attrited")
mean(ensemble_preds == test_set$Status)
```

```
## [1] 0.8805031
```

```r
# listing model results
model_results <- resamples(list(glm = train_glm,
                                lda = train_lda,
                                qda = train_qda,
                                knn = train_knn,
                                rf = train_rf))
summary(model_results)
```

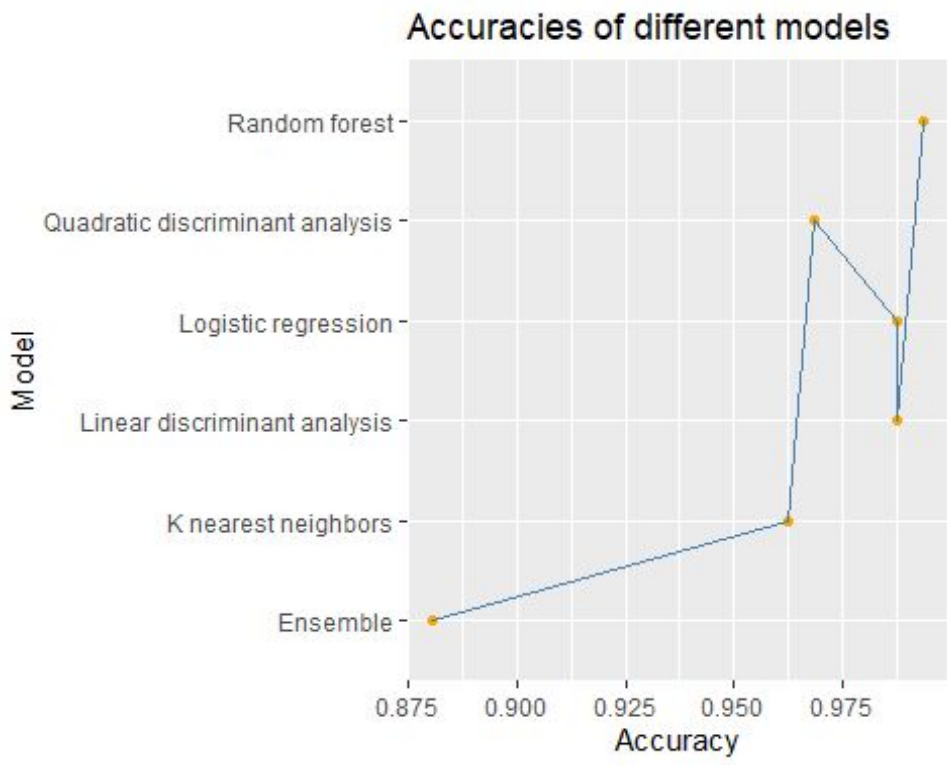
```
##
## Call:
## summary.resamples(object = model_results)
##
## Models: glm, lda, qda, knn, rf
## Number of resamples: 25
##
## Accuracy
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## glm 0.9834025 0.9957983 1.0000000 0.9977992 1.0000000 1.0000000    0
## lda 0.9862385 0.9915966 0.9957447 0.9955547 1.0000000 1.0000000    0
## qda 0.9417040 0.9653680 0.9777778 0.9747081 0.9824561 0.9913043    0
## knn 0.9388646 0.9497717 0.9603524 0.9616795 0.9743590 0.9913420    0
## rf  0.9956140 1.0000000 1.0000000 0.9991376 1.0000000 1.0000000    0
##
## Kappa
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## glm 0.9259601 0.9824506 1.0000000 0.9903036 1.0000000 1.0000000    0
## lda 0.9334148 0.9617406 0.9805947 0.9789744 1.0000000 1.0000000    0
## qda 0.7609072 0.8618582 0.9042638 0.8924232 0.9309091 0.9654914    0
```

```
## knn 0.6758342 0.7486279 0.7908136 0.8024589 0.8554663 0.9616979    0
## rf  0.9799633 1.0000000 1.0000000 0.9963240 1.0000000 1.0000000    0
```

```
# Summarising accuracy from various models
models <- c("Logistic regression", "Linear discriminant analysis", "Quadratic discriminant
analysis", "K nearest neighbors", "Random forest", "Ensemble")

accuracy <- c(mean(glm_preds == test_set$Status),
              mean(lda_preds == test_set$Status),
              mean(qda_preds == test_set$Status),
              mean(knn_preds == test_set$Status),
              mean(rf_preds == test_set$Status),
              mean(ensemble_preds == test_set$Status))
acc_model<- data.frame(Model = models, Accuracy = accuracy)
print(acc_model)
```

```
##                              Model  Accuracy
## 1              Logistic regression 0.9874214
## 2    Linear discriminant analysis 0.9874214
## 3 Quadratic discriminant analysis 0.9685535
## 4              K nearest neighbors 0.9622642
## 5                    Random forest 0.9937107
## 6                         Ensemble 0.8805031
```

```
# Accuracies plot
acc_model %>% ggplot(aes(Accuracy, Model, group=1)) +
  geom_point(color="orange") +
  geom_line(color="steelblue") +
  labs(x = "Accuracy", y = "Model", title = "Accuracies of different models")
```



Accuracy table shows that Random Forest provides the highest accuracy. The confusion matrix printed below also shows that the sensitivity and specificity is almost 1.

```
confusionMatrix(data = rf_preds, reference = test_set$Status)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Attrited Retained
##   Attrited       19        0
##   Retained        1      139
##
##                Accuracy : 0.9937
##                  95% CI : (0.9655, 0.9998)
##     No Information Rate : 0.8742
##     P-Value [Acc > NIR] : 1.245e-08
##
##                   Kappa : 0.9708
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9500
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9929
##              Prevalence : 0.1258
##          Detection Rate : 0.1195
##    Detection Prevalence : 0.1195
##       Balanced Accuracy : 0.9750
##
##        'Positive' Class : Attrited
##
```

Using the tuned rf model, we will arrive at the probability of attrition for each customer in the test set

```
# get customer observations from test set
customers_in_test_set <- customers %>%
  filter(CustomerID %in% row.names(test_set))

rf_prob <- predict(train_rf, customers_in_test_set, type = "prob")

# Compute probabilities
customers_in_test_set$AttritionProbability <- rf_prob$Attrited

rm(test_index, train_set, test_set, train_glm, train_knn, train_lda, train_qda, train_rf,
ensemble)
rm(glm_preds, lda_preds, qda_preds, rf_preds, knn_preds, ensemble_preds, models, accuracy,
acc_model, model_results, rf_prob)
```

---

3. Results:

In the earlier section, we segmented customers with simple scoring and k-means clustering. We identified that 3-cluster solution was interpretable.

We also predicted attrition/retention with an accuracy of .994 using Random Forest method.

Let us examine the predictions along with the customer segments in the test set.

Filter customers with high probability (>=.9) of attrition. Check if the Status shows 'Attrited' and then also see which segment they belong to.

```
res <- customers_in_test_set %>% filter(AttritionProbability>= .9) %>%
  select(CustomerID, RFMScore, C3, Status, AttritionProbability)

table(res$Status)

##
## Attrited
##       18

table(res$RFMScore)

##
##  1  2
## 13  5

table(res$C3)

##
##  1  2  3
##  0  8 10
```

- 100% of customers who have attrited have a probability of .9 or greater
- No customers with RFMScore of 3 (High) have attrited

From 3-Cluster solution:

- 0 cluster 1 customers have attrited (high-recency, high-frequency and high-value)
- 8 cluster 2 customers have attrited (medium-recency, medium-frequency and medium-value)
- 10 cluster 3 customers have attrited (low-recency, low-frequency and low-value)

```
# inspect attrition probability vs status
table(customers_in_test_set$AttritionProbability, customers_in_test_set$Status)

##
##         Attrited Retained
##   0            0      117
##   0.002        0       14
##   0.004        0        2
##   0.006        0        1
##   0.008        0        1
##   0.01         0        2
##   0.022        0        1
##   0.028        0        1
##   0.226        1        0
##   0.79         1        0
##   0.978        1        0
##   0.984        1        0
```

```
##   0.988        1        0
##   0.994        1        0
##   0.996        3        0
##   0.998        4        0
##   1            7        0
```

Table of Customer Status vs Probability of Attrition shows that in almost all cases prediction work. There are of course exceptions (.226 probability customer has also attrited)

Combined information of segments with the probability of attrition would definitely be useful in attending to customers at risk and taking effective measures to retain them.

---

4. Conclusion:

The objective of this project was to segment customers based on Recency, Frequency and Monetary values and predict customer attrition.

- We performed an RFM analysis and segmented customers using simple scoring and k-means clustering.
- We used an ensemble of models to predict customer attrition with RFM as predictors.

We were able to meet the objectives set out for this project.

The dataset that we used did have limitations. It was small to apply machine learning techniques. However, it was good enough to demonstrate the analysis and derive insights from the results.

Many thanks to everyone at Harvard University Online, eDX and Pearson for making this learning possible

---