

# HW5: Predicting Spending Based on Consumer Analysis

Xuyang Ji

2023-03-06

## Table of Contents

0. Project Executive Summary.....	1
I. Data Exploration & Quality Assessment.....	2
II. Data Quality Assessment.....	13
III. KMeans Clustering.....	19
IV. Classification -Decision Tree .....	25
V. Classification - KNN.....	29
VI. Evaluation.....	31
VII. Reflection.....	36

## 0. Project Executive Summary

Customer personality analysis involves collecting and analyzing data about customers, such as their demographics, psychographics, and behaviors, in order to create customer profiles that represent different types of customers. By understanding the unique needs, preferences, and pain points of each customer segment, businesses can modify their products, marketing strategies, and customer service initiatives to better meet the needs of their target customers. This can ultimately lead to improved customer relationships, higher customer satisfaction, and increased revenue and profitability for the business.

The data source used is obtained from [Kaggle](#), comprised of 2,240 observed tuples across 29 attributes. The objective of the study is to divide the target customers on the basis of their significant features which could help the company maintain stronger bond with those high-spending customers in less marketing expenses. Some potential questions to explore are how factors affect consumer spending and are product preferences differ among various segments?

Market segmentation is the process of dividing a broad target market of customers into smaller and more similar groups, and then designing marketing strategies tailored to each group's specific needs and characteristics. **The technique used here are KMeans clustering, in conjunction with PCA for dimension reduction**, which involves analyzing data to automatically identify groups of customers with similar attributes or behaviors. Then, **kNN and Decision Tree Classifiers are applied to the training set with hyperparameter tuning and cross validation**. Finally, the better-performing classifier, Decision Tree Classified, is rebuilt with target variable with two levels instead of three. A

sophisticated evaluation using ***ROC plot, accuracy, precision, and recall metrics is performed on the final model.***

## I. Data Exploration & Quality Assessment

### I.o Preliminary Data Preprocessing

To predict the class label attribute- TotalSpent, of an unseen customer, irrelevant attributes are mutated together or excluded for model-building. Meantime, rows with missing value in the income attributes are removed for further analysis.

New Attributes	Description
Age	Customer's age by calculate diff between birth & 2023
Grad	Customer's education level upon graduation
Married	Customer's marital status
Child	Number of children & teenagers in customer's household
DaysCustomer	Days since customer's enrollment with the company
TotalSpent	Total Amount spent in last 2 years
AcceptCmp	Number of accepted the offer in the last 6 campaign

```
library(readr)
library(tidyverse)

## — Attaching core tidyverse packages ————— tidyverse
2.0.0 —
## ✓ dplyr      1.1.0      ✓ purrr      1.0.1
## ✓ forcats   1.0.0      ✓ stringr    1.5.0
## ✓ ggplot2    3.4.1      ✓ tibble     3.1.8
## ✓ lubridate 1.9.2      ✓ tidyr      1.3.0
## — Conflicts —————
tidyverse_conflicts() —
## ✗ dplyr::filter() masks stats::filter()
## ✗ dplyr::lag()    masks stats::lag()
## ⓘ Use the [8;;http://conflicted.r-lib.org/conflicted-package]8;; to force
all conflicts to become errors

library(dplyr)
library(ggplot2)
library(caret)

## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(readr)
library(ggpubr)
library(psych)

##
## Attaching package: 'psych'
##
## The following objects are masked from 'package:ggplot2':
##
##      %+%, alpha

library(readr)
library(rpart)
library(rattle)

## Loading required package: bitops
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

library(tibble)
library(doSNOW)

## Loading required package: foreach
##
## Attaching package: 'foreach'
##
## The following objects are masked from 'package:purrr':
##
##      accumulate, when
##
## Loading required package: iterators
## Loading required package: snow

library(foreach)
library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at
## https://goo.gl/ve3WBa

library(cluster)
library(cowplot)

##
## Attaching package: 'cowplot'
##
## The following object is masked from 'package:ggpubr':
##
##      get_legend
##
## The following object is masked from 'package:lubridate':
```

```
##
##      stamp

customers <- read_delim("marketing_campaign.csv", delim = "\t", escape_double
= FALSE, trim_ws = TRUE)

## Rows: 2240 Columns: 29
## — Column specification

```

---

```
## Delimiter: "\t"
## chr (3): Education, Marital_Status, Dt_Customer
## dbl (26): ID, Year_Birth, Income, Kidhome, Teenhome, Recency, MntWines,
MntF...
##
## ⓘ Use `spec()` to retrieve the full column specification for this data.
## ⓘ Specify the column types or set `show_col_types = FALSE` to quiet this
message.

customers <- customers %>%
  mutate(Age= 2023-Year_Birth,
         Child= Kidhome + Teenhome,
         AcceptCmp= AcceptedCmp1 + AcceptedCmp2 +
AcceptedCmp3+AcceptedCmp4+AcceptedCmp5+Response,
         DaySCustomer= as.numeric(max(as.Date(Dt_Customer, "%d-%m-%Y"))
- as.Date(Dt_Customer, "%d-%m-%Y")),
         Married=
ifelse(Marital_Status=="Married"|Marital_Status=="Together", 1, 0),
         Grad= ifelse(Education == "2n Cycle" | Education == "Basic", 0, 1),
         TotalSpent= MntMeatProducts+MntFishProducts+MntWines+MntFruits+
MntSweetProducts+ MntGoldProds) %>%
  select(-c(ID, Year_Birth, Kidhome, Teenhome, Dt_Customer, Marital_Status,
AcceptedCmp3,AcceptedCmp4,AcceptedCmp5,AcceptedCmp1,AcceptedCmp2,Response,
Education,Z_Revenue,Z_CostContact))

customers<- data.frame(customers)
missingVal<- customers %>%
  gather(key = "key",value = "value")%>% # gather value from all cols into
columns as key&value
  mutate(na= is.na(value))%>% #create new column na for missing vals
  group_by(key)%>%
  mutate(total= n())%>% #get total observations
  group_by(key,total,na)%>% #grouping each columns,then total col, then na
statement
  summarise(num.na=n())%>%
  mutate(ratioNA= num.na/total*100)

## `summarise()` has grouped output by 'key', 'total'. You can override using
the
## `.groups` argument.
```

```

missingVal %>%
  ggplot()+
  geom_bar(aes(x=reorder(key,desc(ratioNA)),
               y= ratioNA, fill=na),stat="identity",alpha=.7)+
  scale_x_discrete(limits=(missingVal%>%arrange(desc(ratioNA)))$key)+
  scale_fill_manual(name="",values=c("lightblue","darkred"),
    labels = c("PresentVal", "MissingVal")) +
  coord_flip()+
  labs(title = "Percentage of Missing Values", x ='Variable', y = "% of
Missing Values")+
  theme(axis.text=element_text(size=6),
        axis.title=element_text(size=8,face="bold"),
        title =element_text(size=9, face='bold'))

```



```

customers <- customers %>%
  na.omit()

```

### 1.a Preprocessing & Exploration of People Attributes

Looking at the statistics for the people attributes, the distributions of age and income variables are significantly skewed due to outliers and influential points, whereas both plot has a long tail on the right. By converting numeric attributes with less than 8 unique values into factors for analysis, the plot of Grad vs. Income for married and unmarried customers


shown that education does have a positive correlation with income. However, there is no significant difference in income among married and unmarried groups in each education level. Finally, to achieve a roughly symmetric distribution for all variables, the dataset is filtered with  $\text{Income} < \$600000/\text{year}$  and  $\text{Age} \leq 80$  years old.

Variable	Description
Age	Customer's age
Grad	Customer's education level upon graduation
Married	Customer's marital status
Income	Customer's yearly household income
Child	Number of children & tennagers in customer's household
DaysCustomer	Days since customer's enrollment with the company
Recency	Number of days since customer's last purchase
Complain	1 if customer complained in the last 2 years, 0 otherwise

```

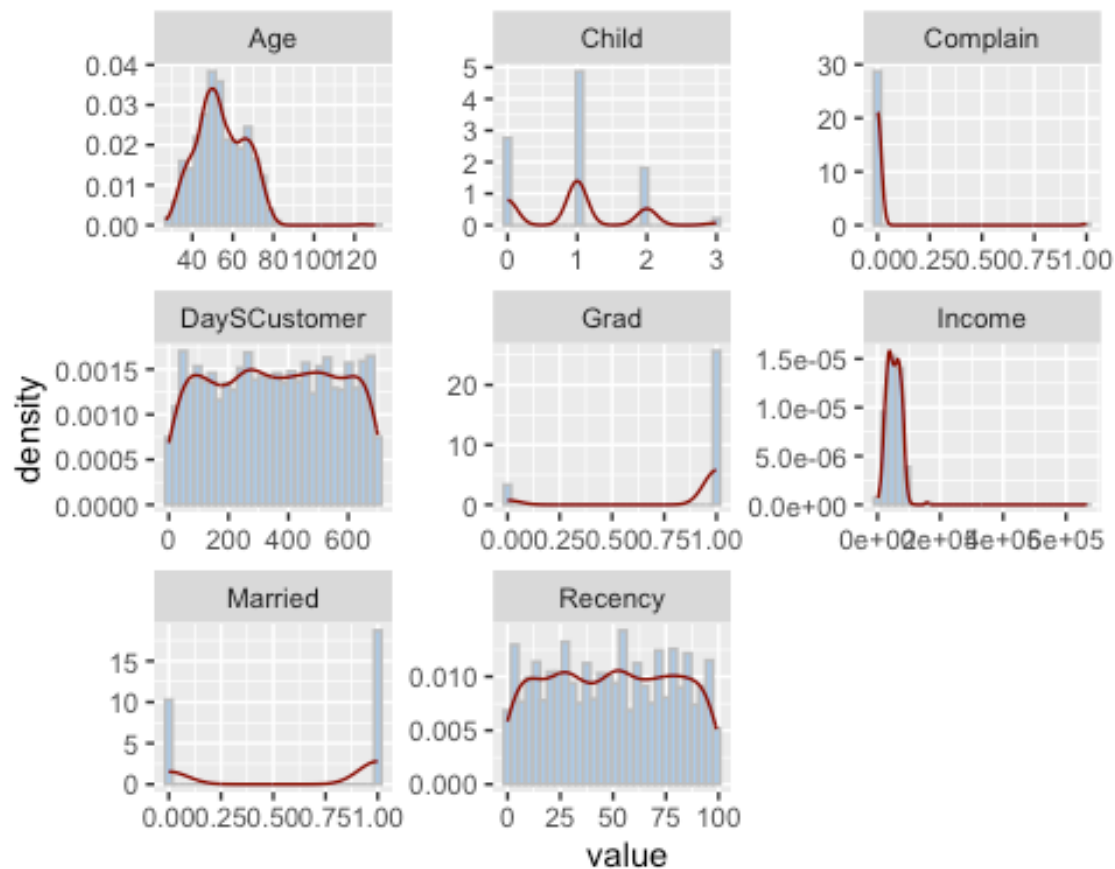
people<- customers %>%
  select(Age, Grad, Married, Income, Child, DaySCustomer, Recency, Complain)
cPlot <- people %>%
  pivot_longer(colnames(people)) %>%
  as.data.frame()
ggplt<- ggplot(cPlot, aes(x=value))+

geom_histogram(aes(y=after_stat(!str2lang("density"))),fill="#adcae6",color=
"grey",bins = 30)+
  geom_density(col="darkred",size=.5)+
  facet_wrap(~name, scales="free")

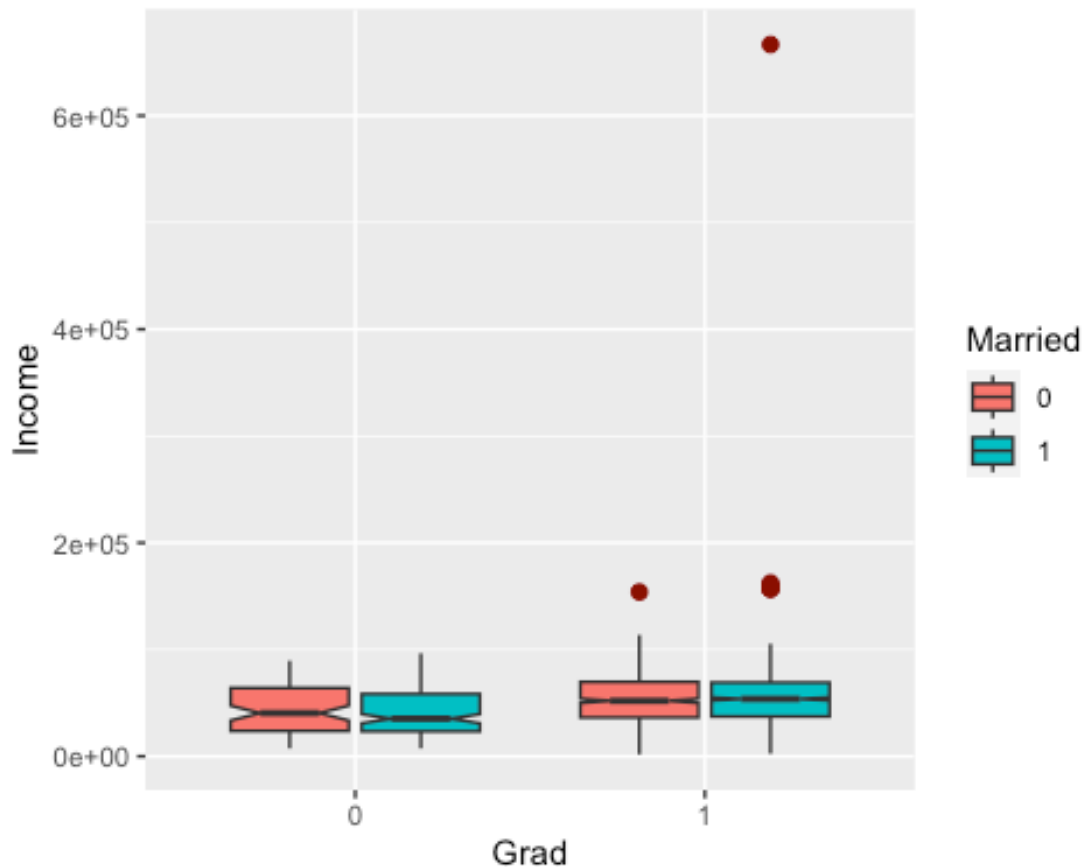
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
##  Please use `linewidth` instead.

ggplt

```



```
# convert less than 8 unique val col into factors
conv <- sapply(customers, function(x) is.numeric(x) && length(unique(x))<8)
customers[conv] <- lapply(customers[conv], as.factor)
ggplot(customers, aes(x= Grad,y=Income,fill=Married))+
  geom_boxplot(outlier.colour = 'darkred',outlier.size=2,notch = T)
```



```
customers <- customers %>%
  filter(Income < 600000) %>%
  filter(Age <= 80)
```

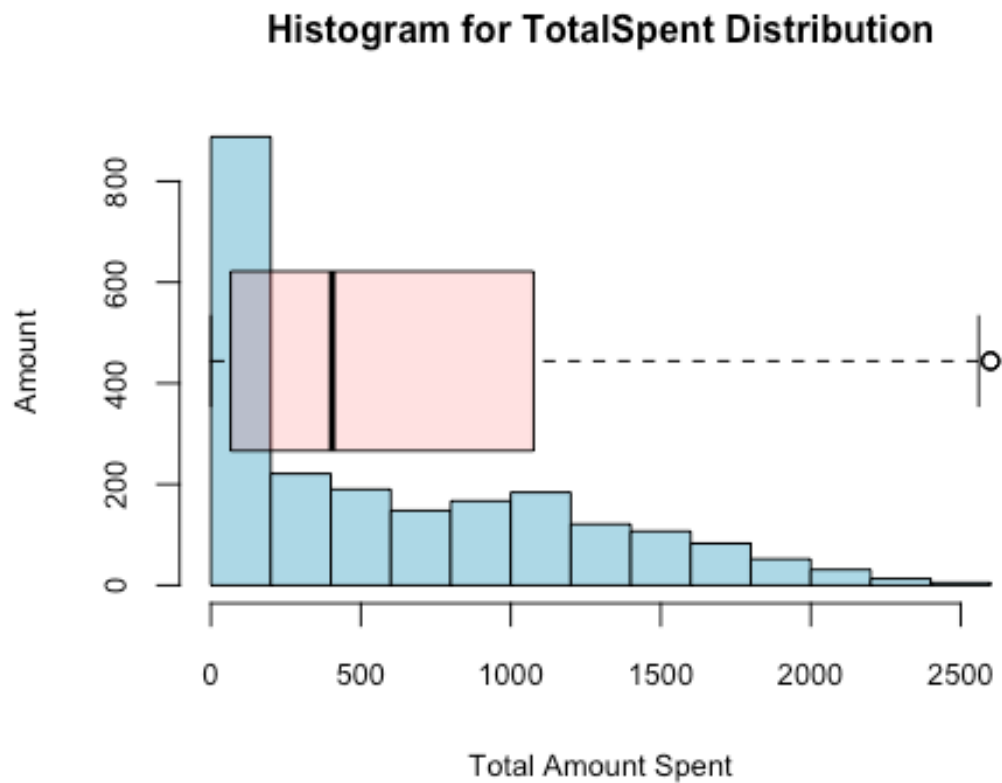
### 1.b Exploration of Products Attributes

The distributions of products attributes are similar in ways that they are all heavily positively skewed, with extremely large values that are considered as outliers. Meantime, by inspecting the standard deviation, the small value means that the data observations of these attributes tend to be very close to the respective mean value. In addition, wine and meat products each has a wider range across distribution compared to other product types.

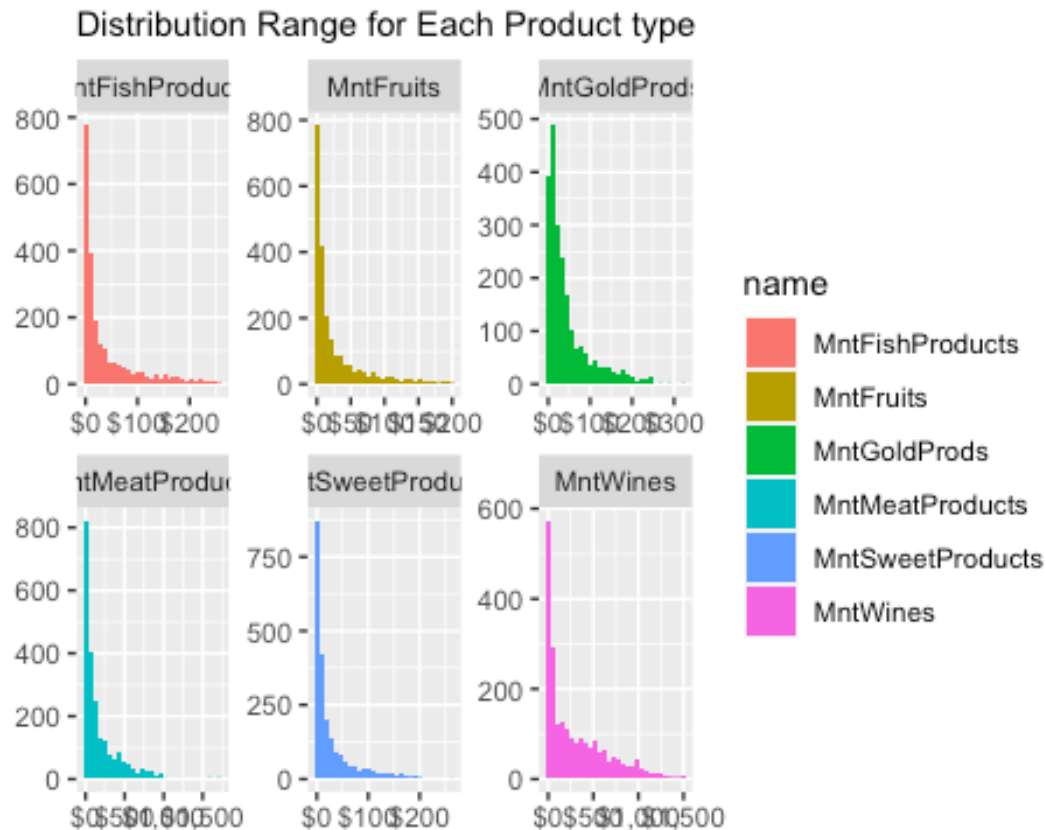
Variable	Description
MntWines	Amount spent on wine in last 2 years
MntFruits	Amount spent on fruits in last 2 years
MntMeatProducts	Amount spent on meat in last 2 years
MntFishProducts	Amount spent on fish in last 2 years
MntSweetProducts	Amount spent on sweets in last 2 years
MntGoldProds	Amount spent on gold in last 2 years
TotalSpent	Total Amount spent in last 2 years



```
hist(customers$TotalSpent,main = "Histogram for TotalSpent
Distribution",breaks=10,cex.main=1,
      cex.lab=.8,cex.axis=.8,xlab = "Total Amount Spent",ylab = "Amount",
border = "black", col = "lightblue")
par(new=TRUE)
boxplot(customers$TotalSpent,horizontal = T,axes=F,col= rgb(1,0,0,alpha=.15))
```



```
customers %>%
  pivot_longer(cols = starts_with("Mnt")) %>%
  select(name,value) %>%
  ggplot(aes(value,fill=name)) +
  geom_histogram(bins = 30)+
  facet_wrap(vars(name),scales = "free")+
  scale_x_continuous(labels = scales::label_dollar()) +
  labs(x = "",
       y = "",
       subtitle = "Distribution Range for Each Product type")
```



### *I.c Exploration of Promotion Attributes*

Variable	Description
----------	-------------

AcceptCmp	Number of accepted the offer in the last 6 campaign
-----------	---

```
plotDF<- data.frame(customers$AcceptCmp,customers$Grad)
```

```
plotDF <- plotDF %>%
```

```
  group_by(customers.Grad)%>%
```

```
  count(customers.AcceptCmp) %>%
```

```
  mutate(Percent=n/sum(n)*100)
```

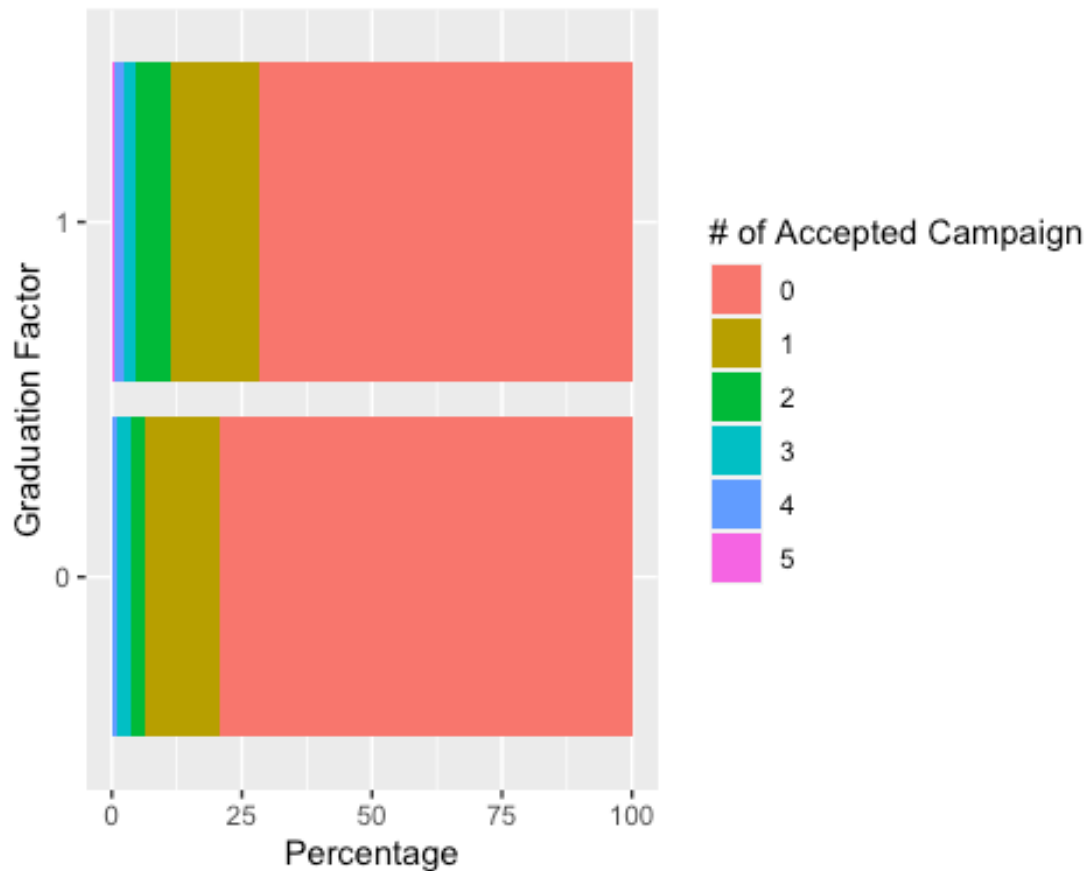
```
ggplot(plotDF, aes(x = customers.Grad, y = Percent, fill =
```

```
customers.AcceptCmp))+
```

```
  geom_bar(stat = "identity")+
```

```
  coord_flip()+
```

```
  labs(x = "Graduation Factor", y = "Percentage",fill = "# of Accepted  
Campaign")
```



#### 1.d Exploration of Place Attributes

Variable	Description
NumDealsPurchases	Number of purchases made with a discount
NumWebPurchases	Number of purchases made through the company's web site
NumCatalogPurchases	Number of purchases made using a catalogue
NumStorePurchases	Number of purchases made directly in stores
NumWebVisitsMonth	Number of visits to company's web site in the last month

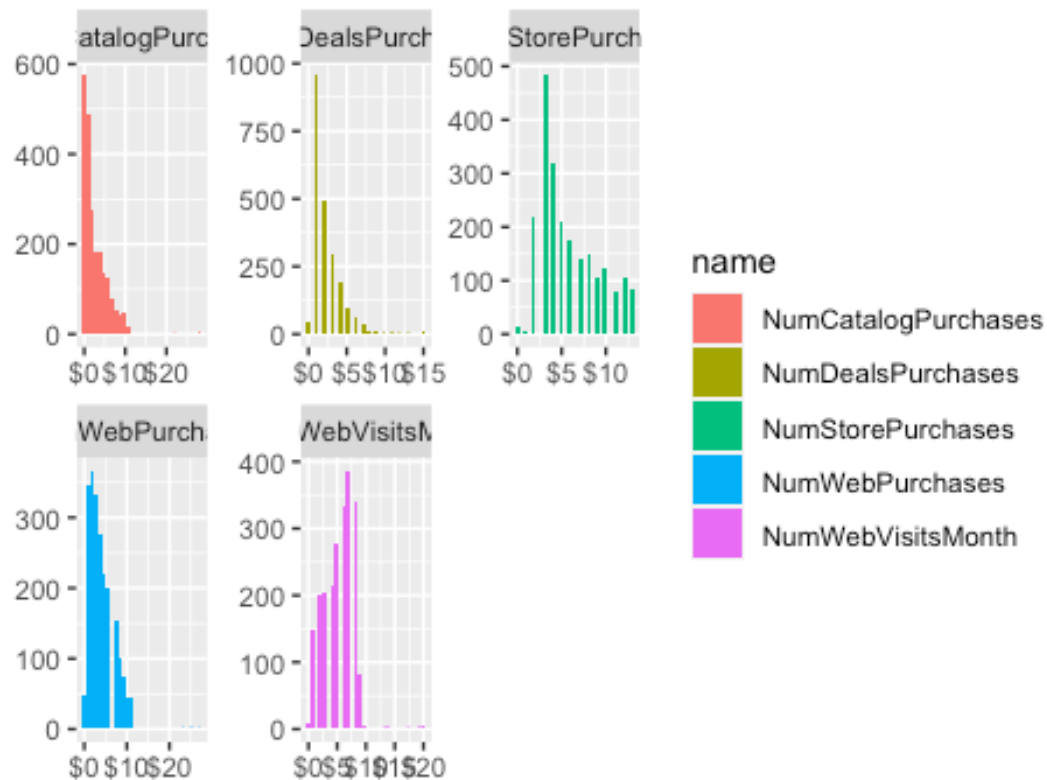
customers %>%

```

pivot_longer(cols = starts_with("Num")) %>%
select(name,value) %>%
ggplot(aes(value,fill=name)) +
geom_histogram(bins = 30)+
facet_wrap(vars(name),scales = "free")+
scale_x_continuous(labels = scales::label_dollar()) +
labs(x = "",
      y = "",
      subtitle = "Distribution Range for Each Place of Purchase")

```

Distribution Range for Each Place of Purchase



#### *I.e Outliers Detection & Removal*

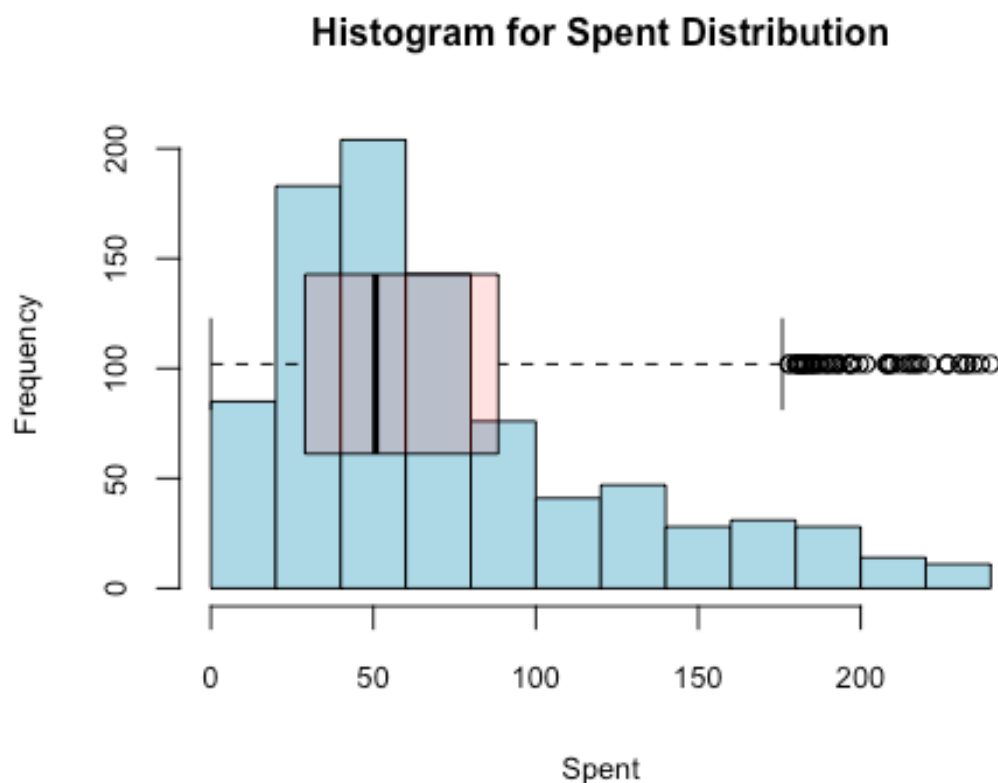
```
detectOutlier<- function(x){
  quantile1<- quantile(x,probs=.25)
  quantile3<- quantile(x,probs=.75) #make the probs large to obtain more
datapoints
  IQR<- quantile3 - quantile1
  x > quantile3 +(IQR*1.5) | x < quantile1 -(IQR*1.5)
}
remove_outlier<- function(df, column){
  for (col in column){
    df<- df[!detectOutlier(df[[col]]),]
  }
  return(df)
}
customer_clean<-
remove_outlier(customers,colnames(customers[,apply(customers,is.numeric)]))
```

For the target attribute, TotalSpent, the boxplot gives a clear indication of the distribution's shape and spread, whereas 50% of the datapoints fall below 57.

```
describe(customer_clean$TotalSpent)
```

```
##      vars    n  mean    sd median trimmed   mad min max range skew kurtosis
se
## X1      1 891 72.62 51.66     57   65.08 38.55    8 240   232 1.21     0.75
1.73

hist(customer_clean$TotalSpent,main = "Histogram for Spent
Distribution",breaks=10,cex.main=1,
      cex.lab=.8,cex.axis=.8,xlab = "Spent",ylab = "Frequency", border =
"black", col = "lightblue")
par(new=TRUE)
boxplot(customer_clean$TotalSpent,horizontal = T,axes=F,col=
rgb(1,0,0,alpha=.15))
```



## II. Data Quality Assessment

### II.a Creating Target Variable Using Binning Methods

```
customer_clean$TotalSpent_bins <- cut(customer_clean$TotalSpent,
                                       breaks = c(8,45,80,240),
                                       include.lowest = TRUE,
                                       right = FALSE,
                                       labels = c("low","medium","high") )
table(customer_clean$TotalSpent_bins)
```

```
##
##    low medium   high
##    318    292    281

customer_clean1<- customer_clean %>%
  select(- TotalSpent)
```

## II.b Data Preparation - Dummy & NZV

```
dummy<- dummyVars(TotalSpent_bins ~., data = customer_clean1)
customer_num <- as.data.frame(predict(dummy, newdata = customer_clean1))

## Warning in model.frame.default(Terms, newdata, na.action = na.action, xlev =
## object$lvls): variable 'TotalSpent_bins' is not a factor

customer_num <- cbind(Spent= customer_clean$TotalSpent_bins, customer_num)
#remove nzv
customer_num <- customer_num[, -nearZeroVar(customer_num)]
```

### ##correlation test

```
corrT <-corr.test(customer_num[2:25],adjust="none")
signTest <- ifelse(corrT$p <0.05, T, F)
colSums(signTest)-1
```

##	Income	Recency	MntWines
MntFruits			
##	17	2	13
16			
##	MntMeatProducts	MntFishProducts	MntSweetProducts
MntGoldProds			
##	16	16	17
15			
##	NumDealsPurchases	NumWebPurchases	NumCatalogPurchases
NumStorePurchases			
##	18	17	13
12			
##	NumWebVisitsMonth	Age	Child.0
Child.1			
##	15	13	13
10			
##	Child.2	AcceptCmp.0	AcceptCmp.1
DaySCustomer			
##	12	9	11
13			
##	Married.0	Married.1	Grad.0
Grad.1			
##	1	1	13
13			

## II.c Principal Component Analysis

Given the large number of attributes, PCA is applied as a method to reduce dimensionality. As an eigenvalue greater than 1 indicates that the PCs account for more variance than accounted by one of the original predictors, those PCs with less than 1 eigenvalue imply that the scores on the PCs would have negative reliability. Using the eigenvalue as a cutoff point in conjunction with an “elbow” shape for determining the number of PCs retained, the optimal number of components are 8 and 11 PCs respectively. Since the goal of the analysis is to examine observations that have a similar pattern, and then distill the variables down to the most important features so that the data is simplified without losing significant traits, data interpretation is the key for this particular dataset. Using the proportion of variance explained criterion, 11 PCs is optimal for a cumulative variance of 84.55% of the original dataset.

```
predictors <- select(customer_num, -c("Spent"))
pca <- prcomp(predictors, scale. = TRUE, center = TRUE)
summary(pca)

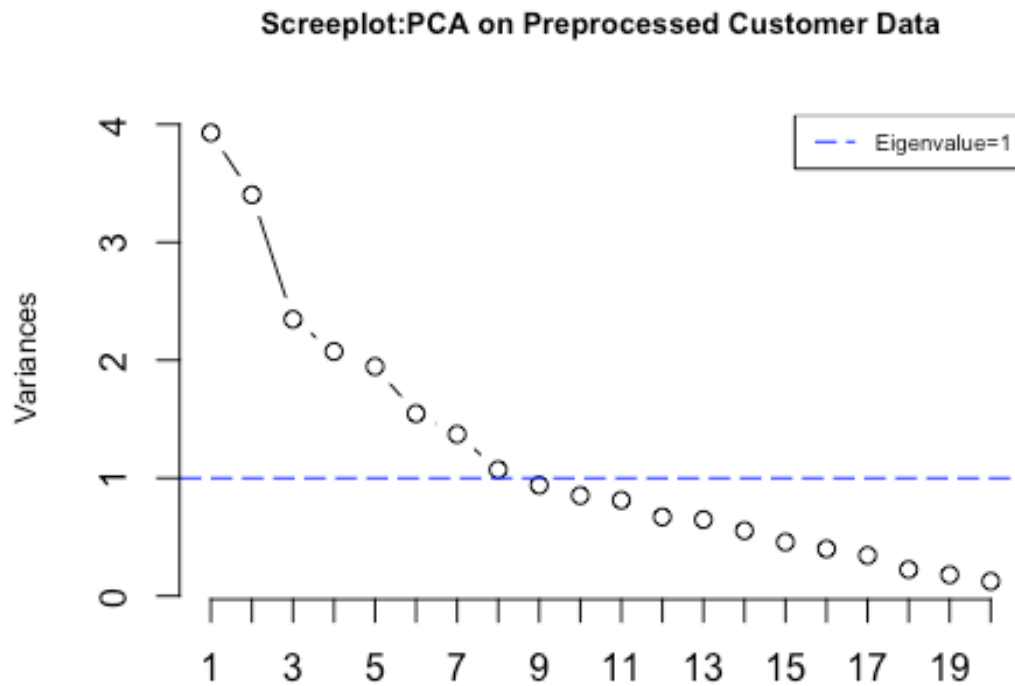
## Importance of components:
##              PC1      PC2      PC3      PC4      PC5      PC6
PC7
## Standard deviation    1.9821 1.8448 1.53181 1.44008 1.39496 1.24370
1.17108
## Proportion of Variance 0.1637 0.1418 0.09777 0.08641 0.08108 0.06445
0.05714
## Cumulative Proportion 0.1637 0.3055 0.40326 0.48967 0.57075 0.63520
0.69234
##              PC8      PC9      PC10      PC11      PC12      PC13
PC14
## Standard deviation    1.03516 0.96949 0.9230 0.90129 0.81875 0.8049
0.7446
## Proportion of Variance 0.04465 0.03916 0.0355 0.03385 0.02793 0.0270
0.0231
## Cumulative Proportion 0.73699 0.77615 0.8116 0.84549 0.87342 0.9004
0.9235
##              PC15      PC16      PC17      PC18      PC19      PC20
PC21
## Standard deviation    0.67637 0.63212 0.58616 0.47354 0.42443 0.3533
0.23386
## Proportion of Variance 0.01906 0.01665 0.01432 0.00934 0.00751 0.0052
0.00228
## Cumulative Proportion 0.94258 0.95923 0.97355 0.98289 0.99040 0.9956
0.99788
##              PC22      PC23      PC24
## Standard deviation    0.22577 1.055e-15 2.216e-16
## Proportion of Variance 0.00212 0.000e+00 0.000e+00
## Cumulative Proportion 1.00000 1.000e+00 1.000e+00

##screeplot - use 11 components
screeplot(pca, npcs = 20, type="line",
```

```

    main="Screeplot:PCA on Preprocessed Customer Data",
    cex.lab=.8,cex.main=0.8)
abline(1,0,col="blue",lty=5)
legend("topright", legend = c("Eigenvalue=1"), col = c("blue"),
      lty = 5,cex = 0.6)

```

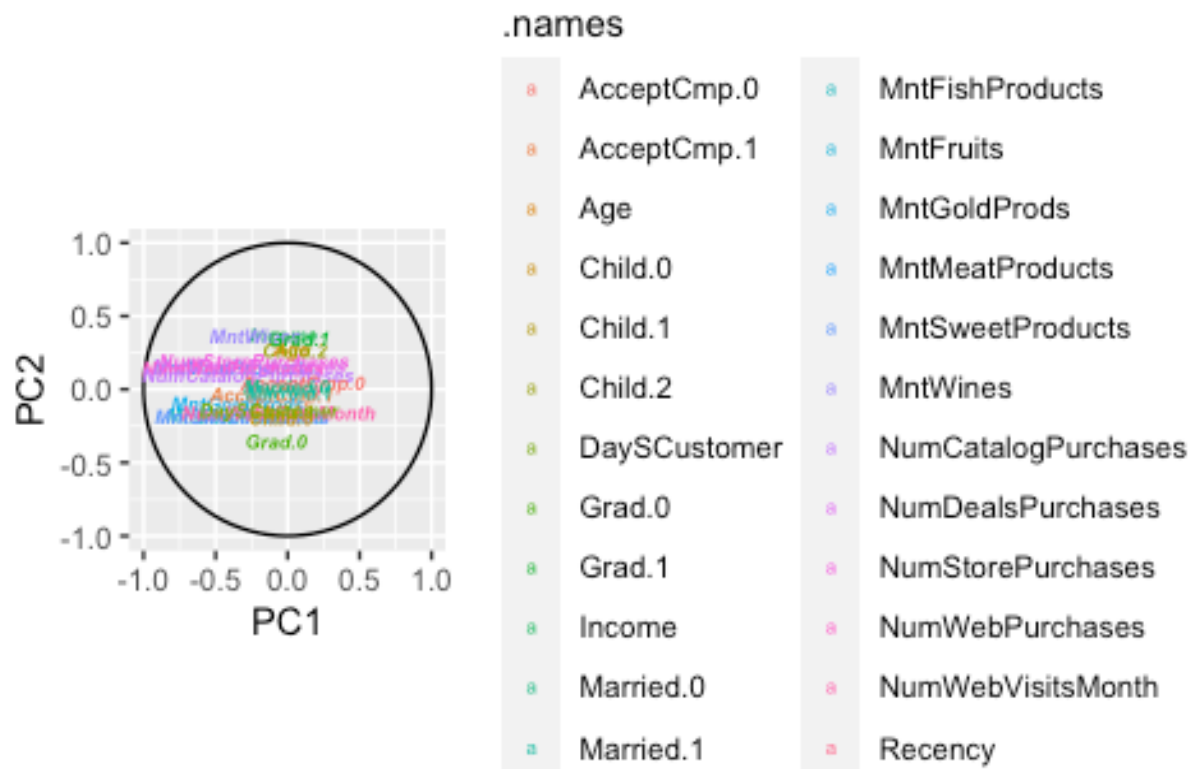


```

PCA_Plot <- function(pcaData){
  theta = seq(0,2*pi,length.out = 100)
  circle = data.frame(x = cos(theta), y = sin(theta))
  p = ggplot(circle,aes(x,y))+ geom_path()
  loadings = data.frame(pcaData$rotation, .names =
row.names(pcaData$rotation))
  p + geom_text(data=loadings, size=2,mapping=aes(x = PC1, y = PC2, label =
.names, colour = .names, fontface="bold.italic")) +
  coord_fixed(ratio=1) + labs(x = "PC1", y = "PC2")
}
PCA_Plot(pca)

```





The Varimax factor rotation is applied to clarify the relationship among the PCA factors. By maximizing the variance shared among selected features, results more discretely represent how features correlated with each principal component. The magnitude and direction of the coefficients for each component represents the importance in calculating the component. As shown below, they are obscure to separate in terms of meaning. For example, PC1 roughly corresponds to product preference of the customer, whereas PC2 is heavily affected by education-related attributes. The PC3 has heavy association with whether the consumer is sensitive to the past campaigns, and PC4 shows household structure and age of the consumer.

```
rawloadings<- pca$rotation[,1:8] %*%diag(pca$sdev,8,8)
rotatedLoading<- varimax(rawloadings)$loadings
print(rotatedLoading,cutoff=.4,sort=T)

##
## Loadings:
##           [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]
## MntFruits    -0.704
## MntFishProducts -0.644
## MntSweetProducts -0.694
## MntGoldProds   -0.517
## Child.0       -0.695
## Grad.0         -0.939
## Grad.1         0.939
```

```

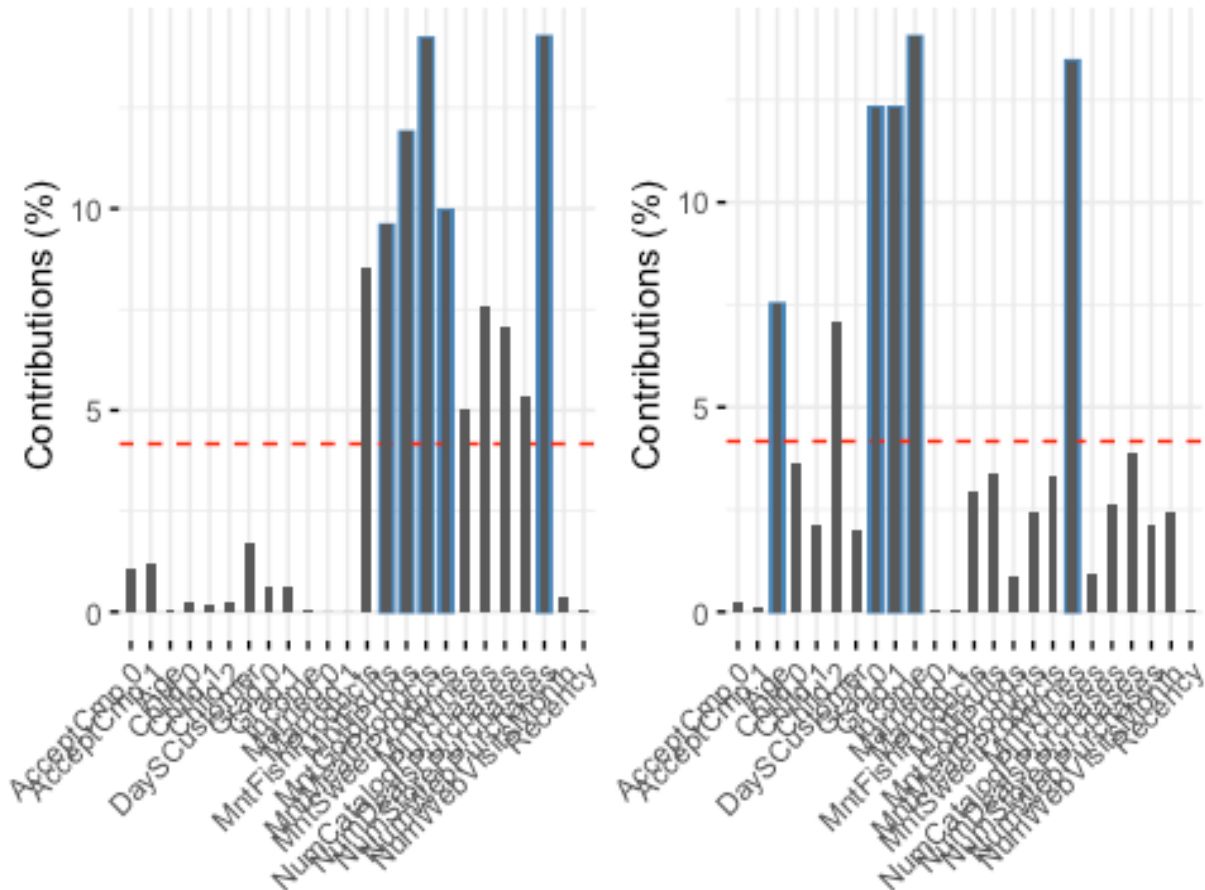
## AcceptCmp.0                -0.927
## AcceptCmp.1                0.886
## Age                        0.579
## Child.1                    -0.946
## Child.2                    0.838
## Married.0                  0.998
## Married.1                  -0.998
## Income                     -0.657
## NumWebVisitsMonth          0.828
## DaySCustomer               0.606
## MntWines                   -0.724
## MntMeatProducts            -0.716
## NumDealsPurchases          -0.701
## NumWebPurchases            -0.802
## NumStorePurchases          -0.645
## Recency                    -
0.725
## NumCatalogPurchases       0.414
0.555
##
##          [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## SS loadings  2.780 2.097 2.227 2.174 2.012 2.016 3.203 1.179
## Proportion Var 0.116 0.087 0.093 0.091 0.084 0.084 0.133 0.049
## Cumulative Var 0.116 0.203 0.296 0.387 0.470 0.554 0.688 0.737

dim1<- fviz_contrib(pca,choice = "var",axes = 1,top = 5,sort.val = "desc")+
  geom_col(width = 0.5)+
  theme(axis.text.x = element_text(angle=45))
dim2<- fviz_contrib(pca,choice = "var",axes = 2,top = 5,sort.val = "desc")+
  geom_col(width = 0.5)+
  theme(axis.text.x = element_text(angle=45))

plot_grid(plotlist = list(dim1,dim2),ncol = 2)

```

## Contribution of variables to Discriminant



```
preProc <- preProcess(predictors, method = c("center", "scale", "pca"), pcaComp
= 11)
customerPC <- predict(preProc, predictors)
customerPC$Spent <- customer_num$Spent
```

### II.d Data Splitting -Train vs.Test

#### ##train-test split

```
set.seed(961)
indexCus<- createDataPartition(y=customerPC$Spent, p=0.85, list = FALSE)
train <- customerPC[indexCus,]
Test <- customerPC[-indexCus,]

trainX<- train%>%
  select(!c(Spent))
```

### III. KMeans Clustering

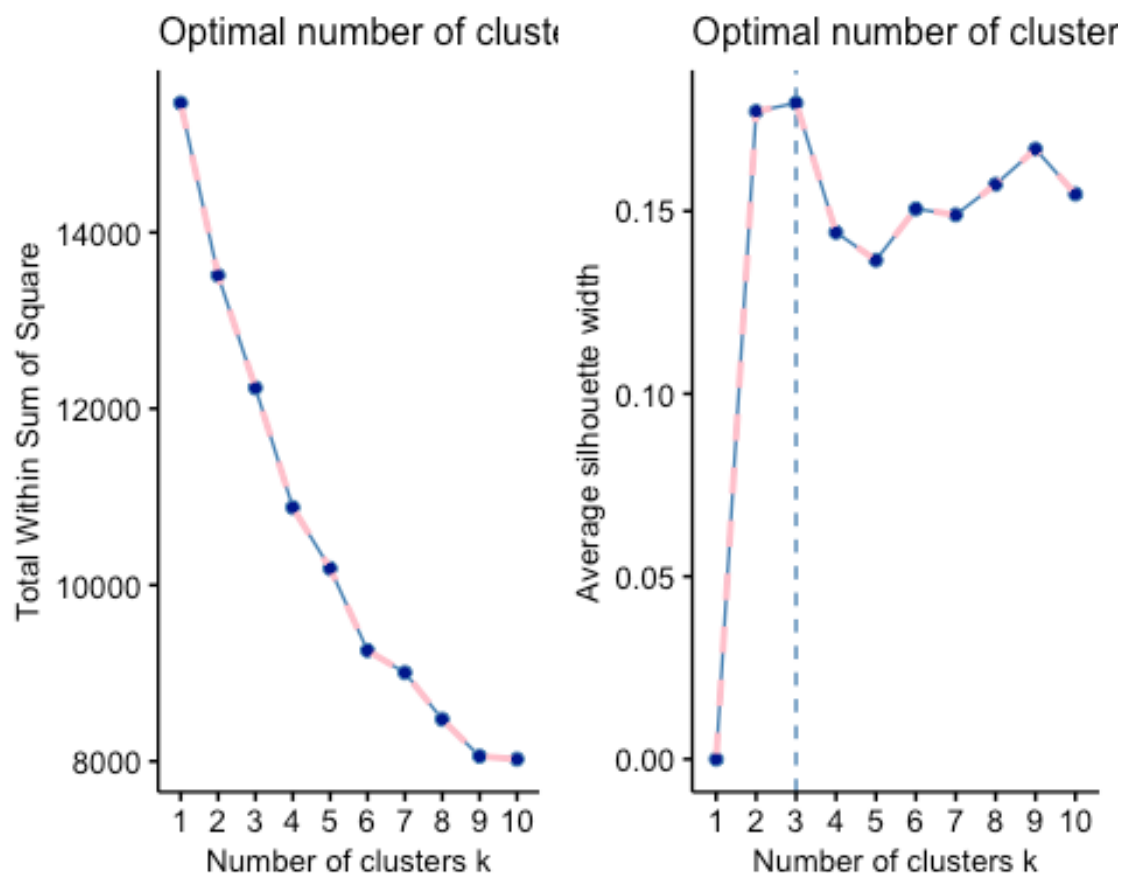
To segment consumers into distinct clusters based on their behaviors, the K-means clustering algorithm can be used. This algorithm groups data into K clusters by iteratively improving the clustering until no further improvement is possible. The elbow method and

silhouette scores indicate that K=3 is an optimal value for the KMeans model. Therefore, the model is fit using K=3 as suggested by the plots.

```
library(cowplot)
wss<- fviz_nbclust(trainX, kmeans, method = "wss")+
  theme(axis.text.x = element_text(size = 10),
        axis.text.y = element_text(size = 10),
        title = element_text(size = 10)) +
  geom_line(aes(group = 1), color = "pink", linetype = "dashed",size = 1) +
  geom_point(group = 1, size = 1, color = "darkblue")

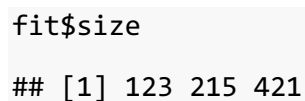
sil<- fviz_nbclust(trainX, kmeans, method = "silhouette")+
  theme(axis.text.x = element_text(size = 10),
        axis.text.y = element_text(size = 10),
        title = element_text(size = 10)) +
  geom_line(aes(group = 1), color = "pink", linetype = "dashed",size = 1) +
  geom_point(group = 1, size = 1, color = "darkblue")

plot_grid(plotlist = list(wss,sil),ncol = 2)
```



Based on the output, the three clusters each with the size of 123- medium spending power, 215 - high spending power, and 421- low spending power observations. Using the `fviz_cluster` function, we can visualize how the clusters are formed. Recall that, PC1 is

```
fit <- kmeans(trainX, centers = 3, nstart = 50)
fviz_cluster(fit, data = trainX)
```

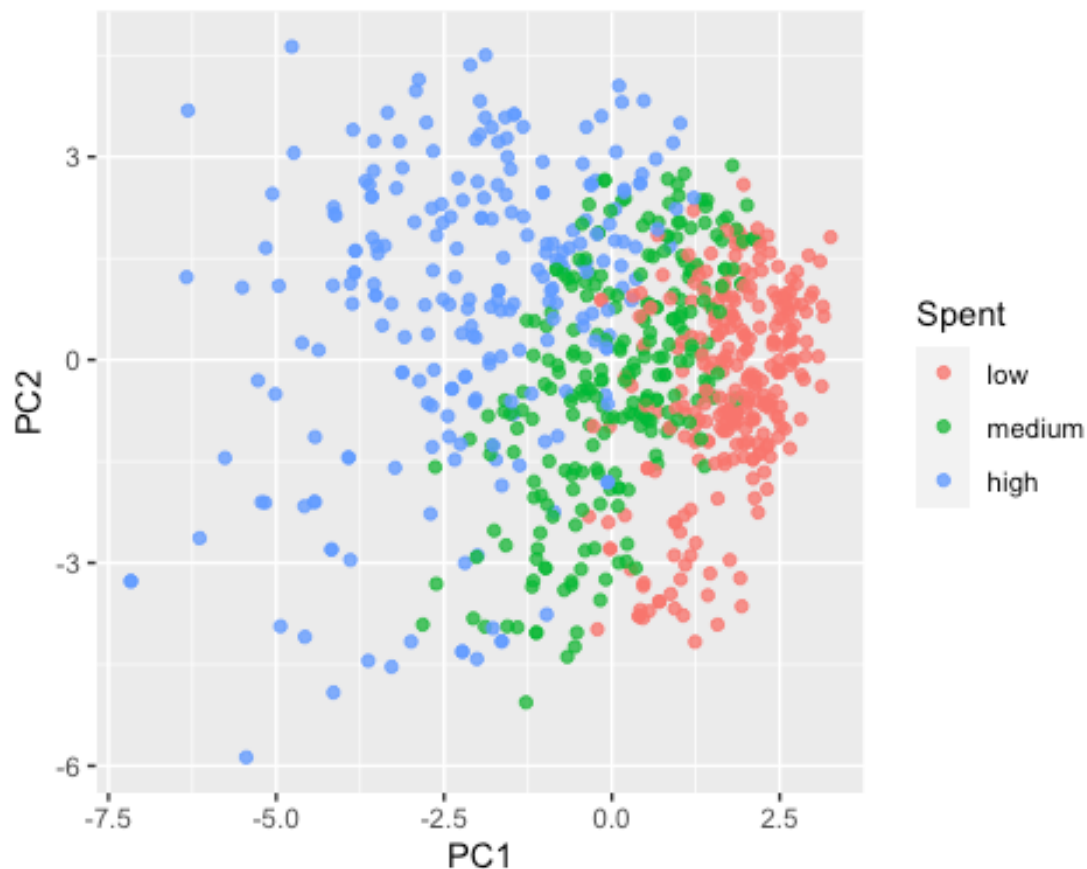


Further, to compare the clusters in the PCA plot, high spending customers tend to have a negative value in PC1, indicating they are more likely to have children in their household, and would prefer to purchase products like meat and wine. In contrast, low spending consumers tend to have positive score in PC1, recall that the strongest coefficients are all negative, meaning they are likely to purchase products like fruit, sweet and fish and possibly have no kids in the household.

```

rotated_data<- as.data.frame(pca$x[indexCus,])
rotated_data$Spent<- train$Spent
rotated_data$Clusters<- as.factor(fit$cluster)
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col =Spent)) +
  geom_point(alpha = 0.8)

```



The plots suggest that income is one of the most significant variables when it comes to predicting spending power, while another important variable is the number of days a customer has been with the company. However, it's worth noting that while income and consumer loyalty can be crucial factors, other factors such as lifestyle factors can also play a significant role in determining spending behavior. Meantime, those high-spending customers, shown as cluster2, are much more deal-sensitive than the rest groups and spend more in wine and meat.

```

trainSegment<- mutate(predictors[indexCus,],cluster=fit$cluster, TotalSpent=
customer_clean$TotalSpent[indexCus])

```

```
count(trainSegment,cluster)
```

```

##  cluster  n
## 1      1 123
## 2      2 215
## 3      3 421

```

```
incomeP<- trainSegment %>% ggplot(aes(Income))+
  geom_histogram(color = "black", fill = "lightblue") +
  facet_wrap(vars(cluster)) +
  geom_vline(aes(xintercept=mean(Income)),color="darkred", linetype="dashed",
size = .7)

daysP<- trainSegment %>% ggplot(aes(DaySCustomer))+
  geom_histogram(color = "black", fill = "purple") +
  facet_wrap(vars(cluster)) +
  geom_vline(aes(xintercept=mean(DaySCustomer)),color="darkred",
linetype="dashed", size = .7)

ageP<- trainSegment %>% ggplot(aes(Age))+
  geom_histogram(color = "black", fill = "pink") +
  facet_wrap(vars(cluster)) +
  geom_vline(aes(xintercept=mean(Age)),color="darkred", linetype="dashed",
size = .7)

plot_grid(plotlist = list(incomeP, daysP, ageP),ncol = 1)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```

library(reshape)

##
## Attaching package: 'reshape'

## The following object is masked from 'package:cowplot':
##
##     stamp

## The following object is masked from 'package:lubridate':
##
##     stamp

## The following object is masked from 'package:dplyr':
##
##     rename

## The following objects are masked from 'package:tidyr':
##
##     expand, smiths

placeP<- trainSegment %>%
  select((cols = starts_with("Num")),cluster) %>%
  melt(id='cluster')%>%
  ggplot(aes(as_factor(cluster), value))+
  geom_boxplot(color = "black", fill = "lightblue")+
  facet_wrap(~variable, ncol = 5)

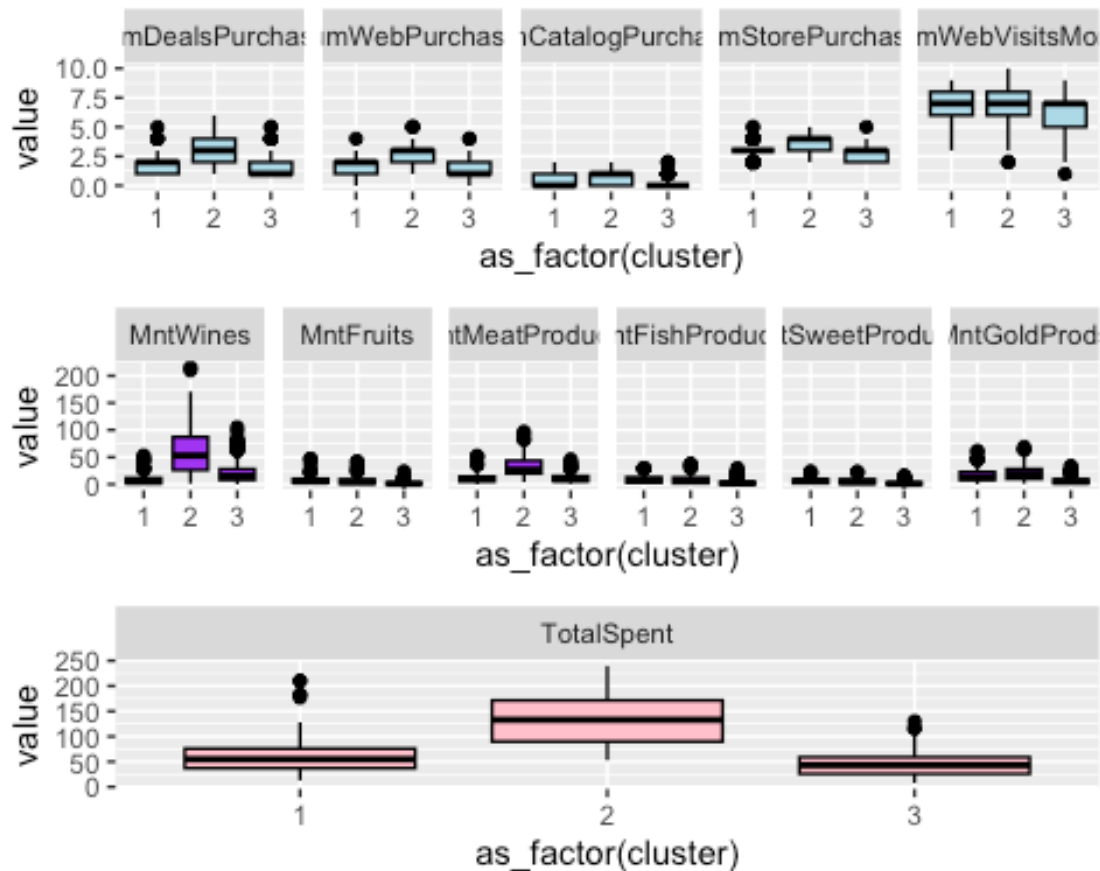
buyP<- trainSegment %>%
  select((cols = starts_with("Mnt")),cluster) %>%
  melt(id='cluster')%>%
  ggplot(aes(as_factor(cluster), value))+
  geom_boxplot(color = "black", fill = "purple")+
  facet_wrap(~variable, ncol = 6)

spentP<- trainSegment %>%
  select(TotalSpent,cluster) %>%
  melt(id='cluster')%>%
  ggplot(aes(as_factor(cluster), value))+
  geom_boxplot(color = "black", fill = "pink")+
  facet_wrap(~variable, ncol = 6)

plot_grid(plotlist = list(placeP, buyP,spentP),ncol = 1)

```





#### IV. Classification -Decision Tree

To predict the class label attribute- TotalSpent, of an unseen customer, the decision tree model is built based on the 11 PCs. Based on the previous step, we have a roughly balanced class distribution. Using 10-fold stratified cross-validation with tuned hyperparameters, the resulting decision tree model on the basis of Gini's impurity index, has an accuracy rate of approximately 82.6% on the test set.

```
print(prop.table(table(train$Spent))) ##Balanced class

##
##      low      medium      high
## 0.3570487 0.3280632 0.3148880

##initialize stratified cross validation
idCategory <- createFolds(train$Spent, k=10, returnTrain = TRUE)
train_control<- trainControl(index=idCategory, method = "cv", number = 10)

##create hyperparameter matrix
maxdepth<- c(3,5,7)
minsplit<- c(20,30,40)
minbucket<- c(5,10,15)
hyperparms= expand.grid(maxdepth=maxdepth,
```

```

minsplit=minsplit,minbucket=minbucket)
#Loop through parms values
library(doParallel)

## Loading required package: parallel

##
## Attaching package: 'parallel'

## The following objects are masked from 'package:snow':
##
##   clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##   clusterExport, clusterMap, clusterSplit, makeCluster, parApply,
##   parCapply, parLapply, parRapply, parSapply, splitIndices,
##   stopCluster

registerDoParallel(cores=4)
results=foreach(i=1:nrow(hyperparms),.combine=rbind)%dopar%{
  d=hyperparms[i,]$maxdepth
  s=hyperparms[i,]$minsplit
  b=hyperparms[i,]$minbucket
  fitDT= train(Spent ~., data=train, method="rpart1SE",
               control=rpart.control(minsplit = s, maxdepth = d, minbucket =
b),
               trControl=train_control)
  pred_train<- predict(fitDT, train)
  accuracy_train<- (confusionMatrix(train$Spent, pred_train))$overall[1]
  pred=predict(fitDT, Test)
  accuracy_test<- (confusionMatrix(Test$Spent, pred))$overall[1]
  node<- nrow(fitDT$finalModel$frame)
  data.frame(Node=node, AccuracyTrain=accuracy_train,
             AccuracyTestn=accuracy_test)
}

hyperparms[which.max(results$AccuracyTestn),]

##   maxdepth minsplit minbucket
## 2         5       20         5

comp<-cbind(hyperparms,results)
head(comp[order(-comp$AccuracyTestn),])

##           maxdepth minsplit minbucket Node AccuracyTrain AccuracyTestn
## Accuracy1         5       20         5   19      0.8498024      0.8257576
## Accuracy2         7       20         5   19      0.8498024      0.8257576
## Accuracy4         5       30         5   19      0.8498024      0.8257576
## Accuracy5         7       30         5   19      0.8498024      0.8257576
## Accuracy7         5       40         5   19      0.8498024      0.8257576
## Accuracy8         7       40         5   19      0.8498024      0.8257576

print(hyperparms[which.max(results$AccuracyTestn),])

```

```

## maxdepth minsplitt minbucket
## 2          5          20          5

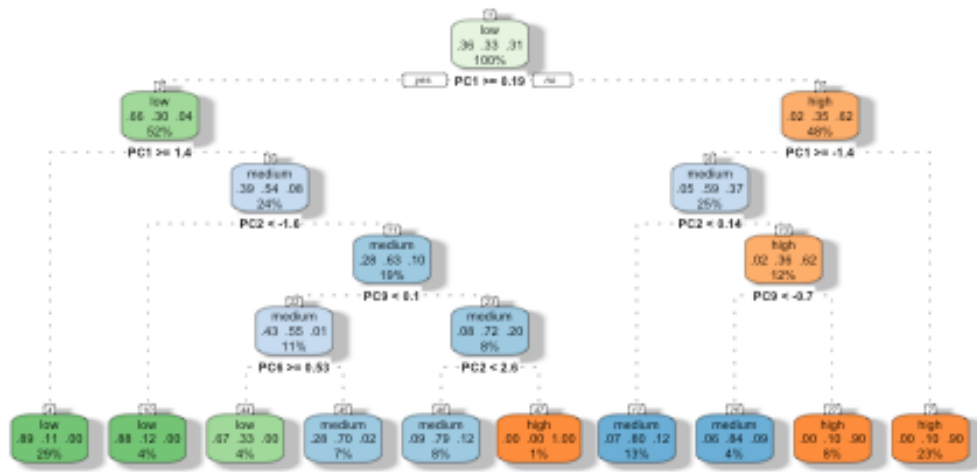
fitBDT= train(Spent ~., data=train, method="rpart1SE",
              control=rpart.control(minsplitt = 20, maxdepth = 5, minbucket = 5),
              trControl=train_control,metric="Accuracy")
predFinal=predict(fitBDT, Test)
accuracy_test<- confusionMatrix(Test$Spent, predFinal)
accuracy_test

## Confusion Matrix and Statistics
##
##              Reference
## Prediction low medium high
## low         41      6    0
## medium      11     27    5
## high         0      1   41
##
## Overall Statistics
##
##              Accuracy : 0.8258
##              95% CI : (0.7501, 0.8862)
## No Information Rate : 0.3939
## P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.738
##
## Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##              Class: low Class: medium Class: high
## Sensitivity          0.7885          0.7941          0.8913
## Specificity          0.9250          0.8367          0.9884
## Pos Pred Value       0.8723          0.6279          0.9762
## Neg Pred Value       0.8706          0.9213          0.9444
## Prevalence           0.3939          0.2576          0.3485
## Detection Rate       0.3106          0.2045          0.3106
## Detection Prevalence 0.3561          0.3258          0.3182
## Balanced Accuracy     0.8567          0.8154          0.9398

fancyRpartPlot(fitBDT$finalModel,main="DecisionTree of Customer Data",caption
= "")

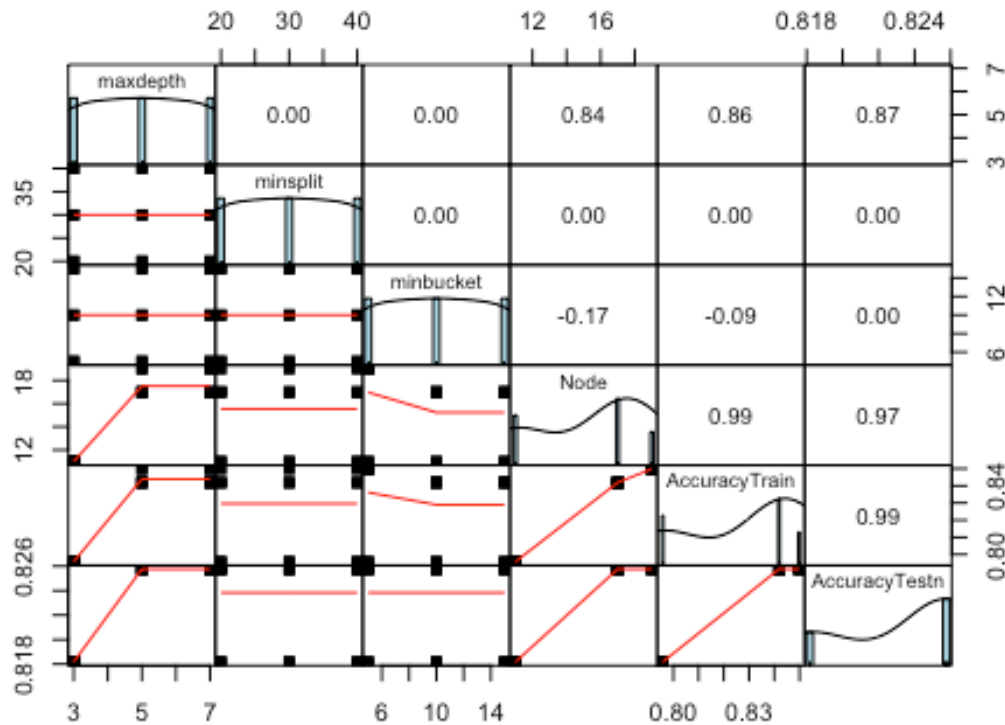
```

## DecisionTree of Customer Data



```
pairs.panels(comp, gap=0, hist.col = "lightblue", density = TRUE, ellipses = FALSE,
             pch = 15, main="Correlation by MaxDepth of Decision Tree", cex=.6)
```

## Correlation by MaxDepth of Decision Tree



## V. Classification - KNN

The following KNN model uses a 10-fold stratified cross-validation trainControl with tuneGrid parameters. The output suggests that the optimal KNN model achieves an accuracy rate of approximately 71.8%, with kmax = 5, distance = 3 and kernel = cos. The Kappa value of 0.57 indicates an average agreement between the predicted and actual labels. Looking at the confusion matrix, the accuracy rate of the KNN classifier on the testing set is 71%, much higher than the no information rate of 39%.

```
tuneGrid <- expand.grid(kmax = 3:7, # test a range of k values 3 to 7
                      kernel = c("cos", "rectangular"), # regular & cosine
                      distance = 1:3) # powers of Minkowski 1 to 3
# tune and fit the model with 10-fold cross validation,
# standardization, and our specialized tune grid
kkn_fit <- train(Spent ~ ., data = train, method = 'kkn',
                trControl = train_control, tuneGrid = tuneGrid)
kkn_fit
```

## k-Nearest Neighbors  
##  
## 759 samples  
## 11 predictor  
## 3 classes: 'low', 'medium', 'high'  
##  
## No pre-processing  
## Resampling: Cross-Validated (10 fold)  
## Summary of sample sizes: 683, 682, 683, 683, 683, 684, ...  
## Resampling results across tuning parameters:  
##

##	kmax	kernel	distance	Accuracy	Kappa
##	3	cos	1	0.6888510	0.5311509
##	3	cos	2	0.6994461	0.5473833
##	3	cos	3	0.6995486	0.5478676
##	3	rectangular	1	0.6914992	0.5355038
##	3	rectangular	2	0.7047610	0.5554632
##	3	rectangular	3	0.6903372	0.5340910
##	4	cos	1	0.6889203	0.5312049
##	4	cos	2	0.6955154	0.5414829
##	4	cos	3	0.7035131	0.5537576
##	4	rectangular	1	0.6914992	0.5355038
##	4	rectangular	2	0.7047610	0.5554632
##	4	rectangular	3	0.6903372	0.5340910
##	5	cos	1	0.6928505	0.5372525
##	5	cos	2	0.6941825	0.5394513
##	5	cos	3	0.7180057	0.5755837
##	5	rectangular	1	0.6914992	0.5355038
##	5	rectangular	2	0.7047610	0.5554632
##	5	rectangular	3	0.6903372	0.5340910
##	6	cos	1	0.6968505	0.5432101

```

##      6      cos      2      0.7022171  0.5514568
##      6      cos      3      0.7167250  0.5735864
##      6  rectangular  1      0.6914992  0.5355038
##      6  rectangular  2      0.7047610  0.5554632
##      6  rectangular  3      0.6903372  0.5340910
##      7      cos      1      0.6994821  0.5471793
##      7      cos      2      0.7035329  0.5534747
##      7      cos      3      0.7167250  0.5735864
##      7  rectangular  1      0.6914992  0.5355038
##      7  rectangular  2      0.7047610  0.5554632
##      7  rectangular  3      0.6903372  0.5340910
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were kmax = 5, distance = 3 and kernel
## = cos.

# Predict
pred_knn <- predict(kknn_fit, Test)

# Generate confusion matrix
confusionMatrix(Test$Spent, pred_knn)

## Confusion Matrix and Statistics
##
##              Reference
## Prediction low medium high
##      low      39      8      0
##      medium  10      28      5
##      high      2      13     27
##
## Overall Statistics
##
##              Accuracy : 0.7121
##              95% CI : (0.6269, 0.7876)
##      No Information Rate : 0.3864
##      P-Value [Acc > NIR] : 3.437e-14
##
##              Kappa : 0.5667
##
##      McNemar's Test P-Value : 0.1229
##
## Statistics by Class:
##
##              Class: low Class: medium Class: high
## Sensitivity      0.7647      0.5714      0.8438
## Specificity      0.9012      0.8193      0.8500
## Pos Pred Value   0.8298      0.6512      0.6429
## Neg Pred Value   0.8588      0.7640      0.9444
## Prevalence       0.3864      0.3712      0.2424
## Detection Rate   0.2955      0.2121      0.2045

```

## Detection Prevalence	0.3561	0.3258	0.3182
## Balanced Accuracy	0.8330	0.6954	0.8469

## VI. Evaluation

Since the previous decision tree model is evaluated based on three classes, the target variable, Spent, is re-binned into two groups, each with 440 in low spending class and 451 tuples in high spending class.

```
customer_clean$TotalSpent_bins2 <- cut(customer_clean$TotalSpent,
                                       breaks = c(8,57,240),
                                       include.lowest = TRUE,
                                       right = FALSE,
                                       labels =c("low","high") )

table(customer_clean$TotalSpent_bins2)

##
##  low high
##  440  451

customer_clean2<- customer_clean %>%
  select(- c(TotalSpent,TotalSpent_bins))

dummy2<- dummyVars(TotalSpent_bins2 ~., data = customer_clean2)
customer_num2 <- as.data.frame(predict(dummy2, newdata = customer_clean2))

## Warning in model.frame.default(Terms, newdata, na.action = na.action, xlev =
## object$lvls): variable 'TotalSpent_bins2' is not a factor

customer_num2 <- cbind(Spent2= customer_clean2$TotalSpent_bins2,
customer_num2)
#remove nzv
customer_num2 <- customer_num2[, -nearZeroVar(customer_num2)]

#PCA
predictors2 <- select(customer_num2,-c("Spent2"))
pca2 <- prcomp(predictors2, scale. = TRUE,center = TRUE)
summary(pca2)

## Importance of components:
##
##              PC1      PC2      PC3      PC4      PC5      PC6
PC7
## Standard deviation    1.9821 1.8448 1.53181 1.44008 1.39496 1.24370
1.17108
## Proportion of Variance 0.1637 0.1418 0.09777 0.08641 0.08108 0.06445
0.05714
## Cumulative Proportion 0.1637 0.3055 0.40326 0.48967 0.57075 0.63520
0.69234
##
##              PC8      PC9      PC10      PC11      PC12      PC13
```

```

PC14
## Standard deviation      1.03516 0.96949 0.9230 0.90129 0.81875 0.8049
0.7446
## Proportion of Variance 0.04465 0.03916 0.0355 0.03385 0.02793 0.0270
0.0231
## Cumulative Proportion  0.73699 0.77615 0.8116 0.84549 0.87342 0.9004
0.9235
##                      PC15      PC16      PC17      PC18      PC19      PC20
PC21
## Standard deviation      0.67637 0.63212 0.58616 0.47354 0.42443 0.3533
0.23386
## Proportion of Variance 0.01906 0.01665 0.01432 0.00934 0.00751 0.0052
0.00228
## Cumulative Proportion  0.94258 0.95923 0.97355 0.98289 0.99040 0.9956
0.99788
##                      PC22      PC23      PC24
## Standard deviation      0.22577 1.055e-15 2.216e-16
## Proportion of Variance 0.00212 0.000e+00 0.000e+00
## Cumulative Proportion  1.00000 1.000e+00 1.000e+00

preProc2 <- preProcess(predictors2, method = c("center", "scale", "pca"),
pcaComp = 11)
customerPC2 <- predict(preProc2, predictors2)
customerPC2$Spent <- customer_num2$Spent2

##train-test split
set.seed(961)
index2<- createDataPartition(y=customerPC2$Spent, p=0.85, list = FALSE)
train2 <- customerPC2[index2,]
Test2 <- customerPC2[-index2,]

trainX2<- train2%>%
  select(!c(Spent))

```

With tuned hyperparameters, the minimum number of items in the parent node that could be split further is set at 20, the maxdepth parameter prevents the tree from growing past a depth of 5, and the minbucket of 10 provides the smallest number of items that are allowed in a terminal node.

```

##initialize stratified cross validation
idCategory2 <- createFolds(train2$Spent, k=10, returnTrain = TRUE)
train_control2<- trainControl(index=idCategory2, method = "cv", number = 10)

##create hyperparameter matrix
maxdepth<- c(3,5,7)
minsplit<- c(20,30,40)
minbucket<- c(5,10,15)
hyperparms= expand.grid(maxdepth=maxdepth,
minsplit=minsplit,minbucket=minbucket)
#Loop through parms values

```



```

library(doParallel)
registerDoParallel(cores=4)
results2=foreach(i=1:nrow(hyperparms),.combine=rbind)%dopar%{
  d=hyperparms[i,]$maxdepth
  s=hyperparms[i,]$minsplit
  b=hyperparms[i,]$minbucket
  fitDT2= train(Spent ~., data=train2, method="rpart1SE",
               control=rpart.control(minsplit = s, maxdepth = d, minbucket =
b),
               trControl=train_control2)
  pred_train<- predict(fitDT2, train2)
  accuracy_train<- (confusionMatrix(train2$Spent, pred_train))$overall[1]
  pred=predict(fitDT2, Test2)
  accuracy_test<- (confusionMatrix(Test2$Spent, pred))$overall[1]
  node<- nrow(fitDT2$finalModel$frame)
  data.frame(Node=node, AccuracyTrain=accuracy_train,
             AccuracyTestn=accuracy_test)
}

hyperparms[which.max(results2$AccuracyTestn),]

##      maxdepth minsplit minbucket
## 11           5        20         10

comp2<-cbind(hyperparms,results)
head(comp2[order(-comp2$AccuracyTestn),])

##      maxdepth minsplit minbucket Node AccuracyTrain AccuracyTestn
## Accuracy1      5        20         5   19      0.8498024      0.8257576
## Accuracy2      7        20         5   19      0.8498024      0.8257576
## Accuracy4      5        30         5   19      0.8498024      0.8257576
## Accuracy5      7        30         5   19      0.8498024      0.8257576
## Accuracy7      5        40         5   19      0.8498024      0.8257576
## Accuracy8      7        40         5   19      0.8498024      0.8257576

```

The final model achieves an accuracy rate of approximately 89.5% on the test set, with a Kappa value of 0.78, indicating the classifier is stronger than expected by chance. The no information rate of 0.52 indicates the accuracy result is higher than NIR at the 0.05 significance level.

Using the Gini index method, PC1 at a value of 0.19 is selected as the first split point with the maximized reduction in impurity. The internal node 2 and 3 further split PC2 at 2.3 and PC1 at 0.29. Looking at the leaf nodes, the rules are:

- IF PC1>=1.2 and PC2 <2.3, THEN class = low
- IF PC1 >=1.2 and PC2 <-0.59, then class= low
- IF PC1 >=1.2, PC2 <-0.59, and PC9 <=0.0048, then class= low
- IF PC1 >=1.2, PC2 <-0.59, and PC9 >0.0048, then class= high

- IF  $PC1 \geq 1.2$  and  $PC2 > 2.3$ , THEN class = high...

```
print(hyperparams[which.max(results2$AccuracyTestn),])

##      maxdepth minsplit minbucket
## 11           5       20         10

fitBDT2= train(Spent ~., data=train2, method="rpart1SE",
               control=rpart.control(minsplit = 20, maxdepth = 5, minbucket =
10),
               trControl=train_control,metric="Accuracy")

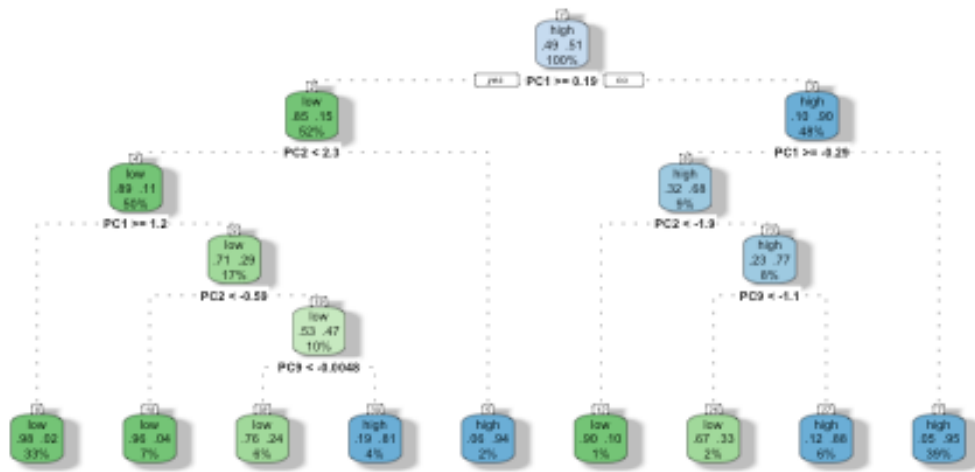
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
trainInfo,
## : There were missing values in resampled performance measures.

predFinal2=predict(fitBDT2, Test2)
accuracy_test2<- confusionMatrix(Test2$Spent, predFinal2)
accuracy_test2

## Confusion Matrix and Statistics
##
##              Reference
## Prediction low high
##          low  61   5
##          high   9  58
##
##              Accuracy : 0.8947
##              95% CI : (0.8297, 0.9412)
##      No Information Rate : 0.5263
##      P-Value [Acc > NIR] : <2e-16
##
##              Kappa : 0.7896
##
##  Mcnemar's Test P-Value : 0.4227
##
##              Sensitivity : 0.8714
##              Specificity : 0.9206
##              Pos Pred Value : 0.9242
##              Neg Pred Value : 0.8657
##              Prevalence : 0.5263
##              Detection Rate : 0.4586
##      Detection Prevalence : 0.4962
##              Balanced Accuracy : 0.8960
##
##              'Positive' Class : low
##

fancyRpartPlot(fitBDT2$finalModel,main="DecisionTree of Customer
Data",caption = "")
```

## DecisionTree of Customer Data



```
##          accuracy_test2$byClass
## Sensitivity          0.8714286
## Specificity          0.9206349
## Pos Pred Value       0.9242424
## Neg Pred Value       0.8656716
## Precision             0.9242424
## Recall                0.8714286
## F1                    0.8970588
## Prevalence            0.5263158
## Detection Rate        0.4586466
## Detection Prevalence  0.4962406
## Balanced Accuracy     0.8960317

Precision<- accuracy_test2$table[1,1]/sum(accuracy_test2$table[1,1:2])
print(Precision)

## [1] 0.9242424

Recall <- accuracy_test2$table[1,1]/sum(accuracy_test2$table[1:2,1])
print(Recall)

## [1] 0.8714286
```

The ROC plot below shown the performance of the binary decision tree classification model in details, the curve that passes through the upper left corner in conjunction with a AUC score of 0.934 indicates the high predicting power of the model on the test set.

```
library(pROC)

## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##      cov, smooth, var

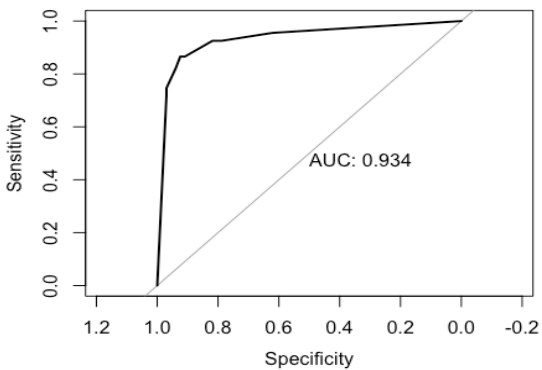
pred_prob<- predict(fitBDT2,Test2,type="prob")
head(pred_prob)

##           low           high
## 16  0.05119454 0.94880546
## 19  0.05119454 0.94880546
## 41  0.98387097 0.01612903
## 45  0.96153846 0.03846154
## 107 0.12244898 0.87755102
## 110 0.05119454 0.94880546

roc_obj<- roc((Test2$Spent),pred_prob[,1])

## Setting levels: control = low, case = high
## Setting direction: controls > cases

plot(roc_obj,print.auc=TRUE)
```



## VII. Reflection

With the integration of statistics, machine learning, and other computing and applied math disciplines, this course introduces me with the fundamental concepts of data science, including techniques and algorithms, and provides hands-on instruction with a powerful toolkit through tutorials and homeworks. I think although the course covers the whole data science process, including gaining an understanding of data, preparing and cleaning data, applying algorithms, and evaluating and communicating results; we still need to grasp the concept learned in lesson in order to face real life problem. Model building is the most interesting concept for me in this course, as I can decide and choose various algorithms such as SVM and decision trees, while automatically discovering structure through algorithms such as clustering and association rule mining.