MM vs CH: Orange Juice Sales Analyses

Nappinnai Jayaseelan

Problem:

The grocery store chain sells two brands of orange juice Citrus Hill (CH) and Minute Maid (MM). MM gets higher margins than CH. Brand Manager is interested in finding out what variables influence a person's probability of buying MM, for increasing the MM sales. Sales Manager is interested in having a predictive model where he can predict the probability of customer purchasing MM.

Problem in Detail:

Brand Manager needs the answers for the following questions basically:

1. What predictor variables influence the purchase of MM?
2. Are all the variables in the dataset effective or are some more effective than others?
3. How confident are you in your recommendations?
4. Based on your analysis what are specific recommendations you have for the brand manager?

Sales manager needs the answers for the following questions basically:

1. Can you provide him a predictive model that can tell him the probability of customers buying MM?
2. How good is the model in its predictions?
3. How confident are you in your recommendations?

Objective: The overall goal is to improve the sales of MM over CH, since MM has higher margin.

Method: This is basically a classic classification problem. For finding which predictors influence the increase in sales of MM orange juice, logistic regression will be the right choice. We can use both logistic regression model and svm model to classify the purchases and find out which model gives the highest level of accuracy. Lets see which model yields the better results.

Initialization

Load the relevant libraries

```
library(knitr)
library(tidyverse)
```

```
## -- Attaching packages ---------------- tidyverse 1.2.1 --
## v ggplot2 3.3.0     v purrr   0.2.5
## v tibble  2.1.1     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
## Warning: package 'ggplot2' was built under R version 3.5.2
## Warning: package 'tibble' was built under R version 3.5.2
## Warning: package 'tidyr' was built under R version 3.5.2
## Warning: package 'dplyr' was built under R version 3.5.2
## -- Conflicts ------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(plotROC)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library("kernlab")
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:purrr':
##
##     cross
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(skimr)
rm(list = ls())
```

Data Load

Load the data from the specified url

```
df <- read.csv(url("http://data.mishra.us/files/OJ.csv"))
dim(df)
```

```
## [1] 1070   18
```

EDA

Lets begin our analyses with EDA.

Summarize the data and observe for NAs

As a first step, we are looking at the structure of the variables and data in various ways. We are also looking at the missing values.

```
summary(df)
```

```
##  Purchase WeekofPurchase     StoreID        PriceCH         PriceMM
##  CH:653   Min.   :227.0   Min.   :1.00   Min.   :1.690   Min.   :1.690
##  MM:417   1st Qu.:240.0   1st Qu.:2.00   1st Qu.:1.790   1st Qu.:1.990
##           Median :257.0   Median :3.00   Median :1.860   Median :2.090
##           Mean   :254.4   Mean   :3.96   Mean   :1.867   Mean   :2.085
##           3rd Qu.:268.0   3rd Qu.:7.00   3rd Qu.:1.990   3rd Qu.:2.180
##           Max.   :278.0   Max.   :7.00   Max.   :2.090   Max.   :2.290
##      DiscCH            DiscMM         SpecialCH        SpecialMM
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.00000   Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :0.05186   Mean   :0.1234   Mean   :0.1477   Mean   :0.1617
##  3rd Qu.:0.00000   3rd Qu.:0.2300   3rd Qu.:0.0000   3rd Qu.:0.0000
```

```
##  Max.   :0.50000   Max.   :0.8000   Max.   :1.0000   Max.   :1.0000
##     LoyalCH          SalePriceMM     SalePriceCH      PriceDiff
##  Min.   :0.000011   Min.   :1.190   Min.   :1.390   Min.   :-0.6700
##  1st Qu.:0.325257   1st Qu.:1.690   1st Qu.:1.750   1st Qu.: 0.0000
##  Median :0.600000   Median :2.090   Median :1.860   Median : 0.2300
##  Mean   :0.565782   Mean   :1.962   Mean   :1.816   Mean   : 0.1465
##  3rd Qu.:0.850873   3rd Qu.:2.130   3rd Qu.:1.890   3rd Qu.: 0.3200
##  Max.   :0.999947   Max.   :2.290   Max.   :2.090   Max.   : 0.6400
##  Store7      PctDiscMM        PctDiscCH       ListPriceDiff
##  No :714   Min.   :0.0000   Min.   :0.00000   Min.   :0.000
##  Yes:356   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.140
##            Median :0.0000   Median :0.00000   Median :0.240
##            Mean   :0.0593   Mean   :0.02731   Mean   :0.218
##            3rd Qu.:0.1127   3rd Qu.:0.00000   3rd Qu.:0.300
##            Max.   :0.4020   Max.   :0.25269   Max.   :0.440
##      STORE
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :2.000
##  Mean   :1.631
##  3rd Qu.:3.000
##  Max.   :4.000
```

```
str(df)
```

```
## 'data.frame':    1070 obs. of  18 variables:
##  $ Purchase      : Factor w/ 2 levels "CH","MM": 1 1 1 2 1 1 1 1 1 1 ...
##  $ WeekofPurchase: int  237 239 245 227 228 230 232 234 235 238 ...
##  $ StoreID       : int  1 1 1 1 7 7 7 7 7 7 ...
##  $ PriceCH       : num  1.75 1.75 1.86 1.69 1.69 1.69 1.69 1.75 1.75 1.75 ...
##  $ PriceMM       : num  1.99 1.99 2.09 1.69 1.69 1.99 1.99 1.99 1.99 1.99 ...
##  $ DiscCH        : num  0 0 0.17 0 0 0 0 0 0 0 ...
##  $ DiscMM        : num  0 0.3 0 0 0 0 0.4 0.4 0.4 0.4 ...
##  $ SpecialCH     : int  0 0 0 0 0 0 1 1 0 0 ...
##  $ SpecialMM     : int  0 1 0 0 0 1 1 0 0 0 ...
##  $ LoyalCH       : num  0.5 0.6 0.68 0.4 0.957 ...
##  $ SalePriceMM   : num  1.99 1.69 2.09 1.69 1.69 1.99 1.59 1.59 1.59 1.59 ...
##  $ SalePriceCH   : num  1.75 1.75 1.69 1.69 1.69 1.69 1.69 1.75 1.75 1.75 ...
##  $ PriceDiff     : num  0.24 -0.06 0.4 0 0 0.3 -0.1 -0.16 -0.16 -0.16 ...
##  $ Store7        : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 2 2 2 2 2 ...
##  $ PctDiscMM     : num  0 0.151 0 0 0 ...
##  $ PctDiscCH     : num  0 0 0.0914 0 0 ...
##  $ ListPriceDiff : num  0.24 0.24 0.23 0 0 0.3 0.3 0.24 0.24 0.24 ...
##  $ STORE         : int  1 1 1 1 0 0 0 0 0 0 ...
```

```
glimpse(df)
```

```
## Observations: 1,070
## Variables: 18
## $ Purchase       <fct> CH, CH, CH, MM, CH, CH, CH, CH, CH, CH, CH, CH,...
## $ WeekofPurchase <int> 237, 239, 245, 227, 228, 230, 232, 234, 235, 23...
## $ StoreID        <int> 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,...
## $ PriceCH        <dbl> 1.75, 1.75, 1.86, 1.69, 1.69, 1.69, 1.69, 1.75,...
## $ PriceMM        <dbl> 1.99, 1.99, 2.09, 1.69, 1.69, 1.99, 1.99, 1.99,...
## $ DiscCH         <dbl> 0.00, 0.00, 0.17, 0.00, 0.00, 0.00, 0.00, 0.00,...
```

```
## $ DiscMM         <dbl> 0.00, 0.30, 0.00, 0.00, 0.00, 0.00, 0.40, 0.40,...
## $ SpecialCH      <int> 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ SpecialMM      <int> 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,...
## $ LoyalCH        <dbl> 0.500000, 0.600000, 0.680000, 0.400000, 0.95653...
## $ SalePriceMM    <dbl> 1.99, 1.69, 2.09, 1.69, 1.69, 1.99, 1.59, 1.59,...
## $ SalePriceCH    <dbl> 1.75, 1.75, 1.69, 1.69, 1.69, 1.69, 1.69, 1.75,...
## $ PriceDiff      <dbl> 0.24, -0.06, 0.40, 0.00, 0.00, 0.30, -0.10, -0....
## $ Store7         <fct> No, No, No, No, Yes, Yes, Yes, Yes, Yes, Yes, Y...
## $ PctDiscMM      <dbl> 0.000000, 0.150754, 0.000000, 0.000000, 0.00000...
## $ PctDiscCH      <dbl> 0.000000, 0.000000, 0.091398, 0.000000, 0.00000...
## $ ListPriceDiff  <dbl> 0.24, 0.24, 0.23, 0.00, 0.00, 0.30, 0.30, 0.24,...
## $ STORE          <int> 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
```

```
head(df)
```

```
##   Purchase WeekofPurchase StoreID PriceCH PriceMM DiscCH DiscMM SpecialCH
## 1       CH            237       1    1.75    1.99   0.00    0.0         0
## 2       CH            239       1    1.75    1.99   0.00    0.3         0
## 3       CH            245       1    1.86    2.09   0.17    0.0         0
## 4       MM            227       1    1.69    1.69   0.00    0.0         0
## 5       CH            228       7    1.69    1.69   0.00    0.0         0
## 6       CH            230       7    1.69    1.99   0.00    0.0         0
##   SpecialMM  LoyalCH SalePriceMM SalePriceCH PriceDiff Store7 PctDiscMM
## 1         0 0.500000        1.99        1.75      0.24     No  0.000000
## 2         1 0.600000        1.69        1.75     -0.06     No  0.150754
## 3         0 0.680000        2.09        1.69      0.40     No  0.000000
## 4         0 0.400000        1.69        1.69      0.00     No  0.000000
## 5         0 0.956535        1.69        1.69      0.00    Yes  0.000000
## 6         1 0.965228        1.99        1.69      0.30    Yes  0.000000
##   PctDiscCH ListPriceDiff STORE
## 1  0.000000          0.24     1
## 2  0.000000          0.24     1
## 3  0.091398          0.23     1
## 4  0.000000          0.00     1
## 5  0.000000          0.00     0
## 6  0.000000          0.30     0
```

```
#skim(df)
```

Missing Values

skim() command shows that there are no missing values. Hence the process of imputation is not needed.

Note: While creating the pdf, it is not allowing me to create pdf with skim command's output. Hence, commented out that line of code.

Null Values

There are no null values in the data.

Observing Data

Some basic observation of data.

```
glimpse(df)
```

```
## Observations: 1,070
```

```
## Variables: 18
## $ Purchase       <fct> CH, CH, CH, MM, CH, CH, CH, CH, CH, CH, CH, CH,...
## $ WeekofPurchase <int> 237, 239, 245, 227, 228, 230, 232, 234, 235, 23...
## $ StoreID        <int> 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,...
## $ PriceCH        <dbl> 1.75, 1.75, 1.86, 1.69, 1.69, 1.69, 1.69, 1.75,...
## $ PriceMM        <dbl> 1.99, 1.99, 2.09, 1.69, 1.69, 1.99, 1.99, 1.99,...
## $ DiscCH         <dbl> 0.00, 0.00, 0.17, 0.00, 0.00, 0.00, 0.00, 0.00,...
## $ DiscMM         <dbl> 0.00, 0.30, 0.00, 0.00, 0.00, 0.00, 0.40, 0.40,...
## $ SpecialCH      <int> 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ SpecialMM      <int> 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,...
## $ LoyalCH        <dbl> 0.500000, 0.600000, 0.680000, 0.400000, 0.95653...
## $ SalePriceMM    <dbl> 1.99, 1.69, 2.09, 1.69, 1.69, 1.99, 1.59, 1.59,...
## $ SalePriceCH    <dbl> 1.75, 1.75, 1.69, 1.69, 1.69, 1.69, 1.69, 1.75,...
## $ PriceDiff      <dbl> 0.24, -0.06, 0.40, 0.00, 0.00, 0.30, -0.10, -0....
## $ Store7         <fct> No, No, No, No, Yes, Yes, Yes, Yes, Yes, Yes, Y...
## $ PctDiscMM      <dbl> 0.000000, 0.150754, 0.000000, 0.000000, 0.00000...
## $ PctDiscCH      <dbl> 0.000000, 0.000000, 0.091398, 0.000000, 0.00000...
## $ ListPriceDiff  <dbl> 0.24, 0.24, 0.23, 0.00, 0.00, 0.30, 0.30, 0.24,...
## $ STORE          <int> 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
```

```
#Purchase - A factor with levels CH and MM indicating whether the customer purchased Citrus Hill or Min
#WeekofPurchase - Week of purchase. Here week 227 is week 1 of a year (i.e., January first week)

#StoreID - Store ID

#PriceCH - Price charged for CH. Also called List Price for CH
#PriceMM - Price charged for MM. Also called List Price for MM
#DiscCH - Discount offered for CH
#DiscCH - Discount offered for MM

#SpecialCH - Indicator of special on CH. Special can be a free gift, loyalty points etc.
#SpecialMM - Indicator of special on MM. Special can be a free gift, loyalty points etc.
#LoyalCH - Customer brand loyalty for CH. That is, probability to buy CH (over MM) based on prior purch

#SalePriceMM - Sale price for MM. This is the difference between the list price and discount.
#SalePriceCH - Sale price for CH. This is the difference between the list price and discount.
##PriceCH - DiscCH = SalePriceCH
##PriceMM - DiscMM = SalePriceMM

#PriceDiff - Sale price of MM less sale price of CH
##PriceDiff = SalePriceMM - SalePriceCH

#Store7 - A factor with levels No and Yes indicating whether the sale is at Store 7
##StoreID has this information already

#PctDiscMM - Percentage discount for MM
##DiscMM/PriceMM = PctDiscMM

#PctDiscCH - Percentage discount for CH
##DiscCH/PriceCH = PctDiscCH

#ListPriceDiff - List price of MM less list price of CH
##PriceMM - PriceCH = ListPriceDiff
```

```
#STORE - Which of 5 possible stores the sale occurred at
```

Checking for zero variance

```
nearZeroVar(df, names = T)
```

```
## character(0)
```

There is no near zero variance in any of the features.

Checking for factorization

```
#StoreID
table(df$StoreID)
```

```
##
##   1   2   3   4   7
## 157 222 196 139 356
```

```
#  1   2   3   4   7
#157 222 196 139 356
#There are 5 store ids; can convert it into factors
df$StoreID <- as.factor(df$StoreID)

#STORE
table(df$STORE)
```

```
##
##   0   1   2   3   4
## 356 157 222 196 139
```

```
#  0   1   2   3   4
#356 157 222 196 139
df$STORE <- as.factor(df$STORE)

#SpecialCH, SpecialMM are indicators - can convert into factors
table(df$SpecialCH)
```

```
##
##   0   1
## 912 158
```

```
#  0   1
#912 158
df$SpecialCH <- as.factor(df$SpecialCH)

table(df$SpecialMM)
```

```
##
##   0   1
## 897 173
```

```
#  0   1
#897 173
df$SpecialMM <- as.factor(df$SpecialMM)
```

```
#Store7
table(df$Store7)
```

```
##
##  No Yes
## 714 356
```

```
# No Yes
#714 356

colnames(df)
```

```
##  [1] "Purchase"       "WeekofPurchase" "StoreID"        "PriceCH"
##  [5] "PriceMM"        "DiscCH"         "DiscMM"         "SpecialCH"
##  [9] "SpecialMM"      "LoyalCH"        "SalePriceMM"    "SalePriceCH"
## [13] "PriceDiff"      "Store7"         "PctDiscMM"      "PctDiscCH"
## [17] "ListPriceDiff"  "STORE"
```

The categorical values are converted into factors.

Different Correlation Plots

```
#Finding out the correlation among the numeric features
cor(df[,unlist(lapply(df, is.numeric))])
```

```
##                 WeekofPurchase     PriceCH      PriceMM      DiscCH
## WeekofPurchase     1.00000000  0.70432413  0.576872269  0.36572228
## PriceCH            0.70432413  1.00000000  0.616401748  0.15190000
## PriceMM            0.57687227  0.61640175  1.000000000  0.06520644
## DiscCH             0.36572228  0.15190000  0.065206435  1.00000000
## DiscMM             0.24233415  0.11631025 -0.001246148  0.01803525
## LoyalCH            0.19289722  0.07779263  0.115569558  0.13940028
## SalePriceMM        0.10171874  0.22938272  0.532858673  0.01941554
## SalePriceCH        0.20125614  0.58671585  0.384941275 -0.71127380
## PriceDiff         -0.01160974 -0.09633508  0.292594396  0.39361535
## PctDiscMM          0.22353257  0.09915740 -0.021747405  0.01471802
## PctDiscCH          0.35504707  0.13460070  0.059963526  0.99902246
## ListPriceDiff      0.05303849 -0.17793470  0.665186965 -0.06255059
##                       DiscMM     LoyalCH SalePriceMM SalePriceCH
## WeekofPurchase  0.242334146  0.19289722  0.10171874  0.20125614
## PriceCH         0.116310246  0.07779263  0.22938272  0.58671585
## PriceMM        -0.001246148  0.11556956  0.53285867  0.38494127
## DiscCH          0.018035253  0.13940028  0.01941554 -0.71127380
## DiscMM          1.000000000 -0.02029164 -0.84686762  0.06793979
## LoyalCH        -0.020291637  1.00000000  0.07863126 -0.05888708
## SalePriceMM    -0.846867615  0.07863126  1.00000000  0.14722240
## SalePriceCH     0.067939786 -0.05888708  0.14722240  1.00000000
## PriceDiff      -0.823907970  0.10426083  0.85279789 -0.39099950
## PctDiscMM       0.998793158 -0.02246037 -0.85674903  0.05845905
## PctDiscCH       0.018521069  0.13868388  0.01621623 -0.72277560
## ListPriceDiff  -0.111847688  0.07065930  0.44839527 -0.07529368
##                  PriceDiff   PctDiscMM   PctDiscCH ListPriceDiff
## WeekofPurchase -0.01160974  0.22353257  0.35504707    0.05303849
## PriceCH        -0.09633508  0.09915740  0.13460070   -0.17793470
## PriceMM         0.29259440 -0.02174741  0.05996353    0.66518696
```

```
## DiscCH          0.39361535  0.01471802  0.99902246   -0.06255059
## DiscMM         -0.82390797  0.99879316  0.01852107   -0.11184769
## LoyalCH         0.10426083 -0.02246037  0.13868388    0.07065930
## SalePriceMM     0.85279789 -0.85674903  0.01621623    0.44839527
## SalePriceCH    -0.39099950  0.05845905 -0.72277560   -0.07529368
## PriceDiff       1.00000000 -0.82809715  0.39671119    0.45700011
## PctDiscMM      -0.82809715  1.00000000  0.01531748   -0.12120275
## PctDiscCH       0.39671119  0.01531748  1.00000000   -0.05269863
## ListPriceDiff   0.45700011 -0.12120275 -0.05269863    1.00000000
```

```
str(lapply(df, is.numeric))
```

```
## List of 18
##  $ Purchase      : logi FALSE
##  $ WeekofPurchase: logi TRUE
##  $ StoreID       : logi FALSE
##  $ PriceCH       : logi TRUE
##  $ PriceMM       : logi TRUE
##  $ DiscCH        : logi TRUE
##  $ DiscMM        : logi TRUE
##  $ SpecialCH     : logi FALSE
##  $ SpecialMM     : logi FALSE
##  $ LoyalCH       : logi TRUE
##  $ SalePriceMM   : logi TRUE
##  $ SalePriceCH   : logi TRUE
##  $ PriceDiff     : logi TRUE
##  $ Store7        : logi FALSE
##  $ PctDiscMM     : logi TRUE
##  $ PctDiscCH     : logi TRUE
##  $ ListPriceDiff : logi TRUE
##  $ STORE         : logi FALSE
```

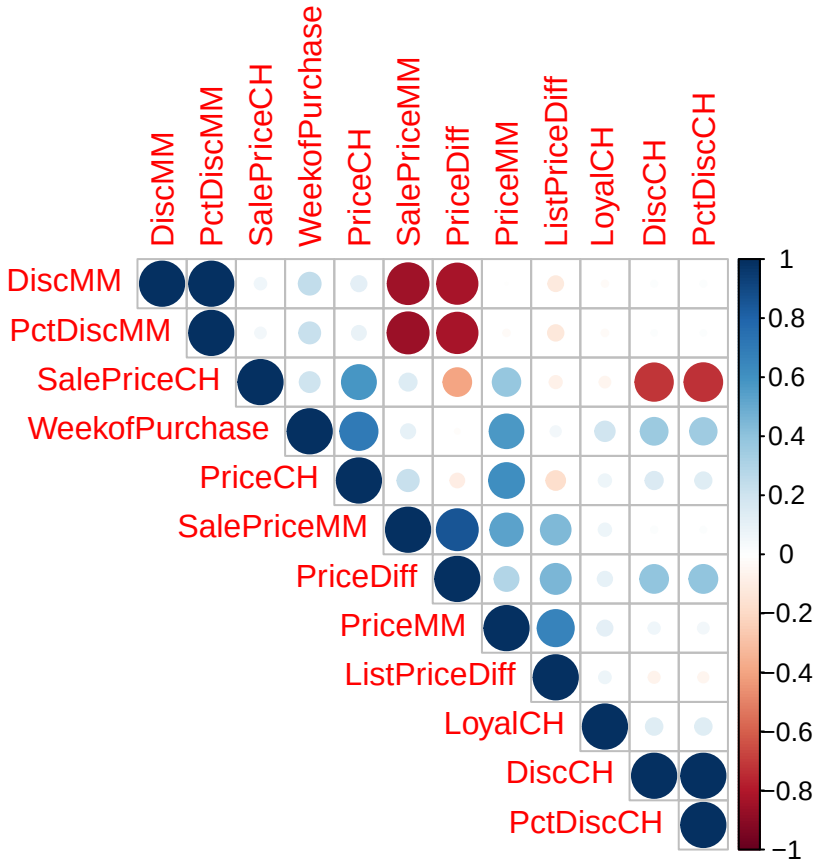
```
str(unlist(lapply(df, is.numeric)))
```

```
##  Named logi [1:18] FALSE TRUE FALSE TRUE TRUE TRUE ...
##  - attr(*, "names")= chr [1:18] "Purchase" "WeekofPurchase" "StoreID" "PriceCH" ...
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
matx <- cor(df[,unlist(lapply(df, is.numeric))])
corrplot(matx, type="upper", order="hclust")
```




```
library(psych)
```

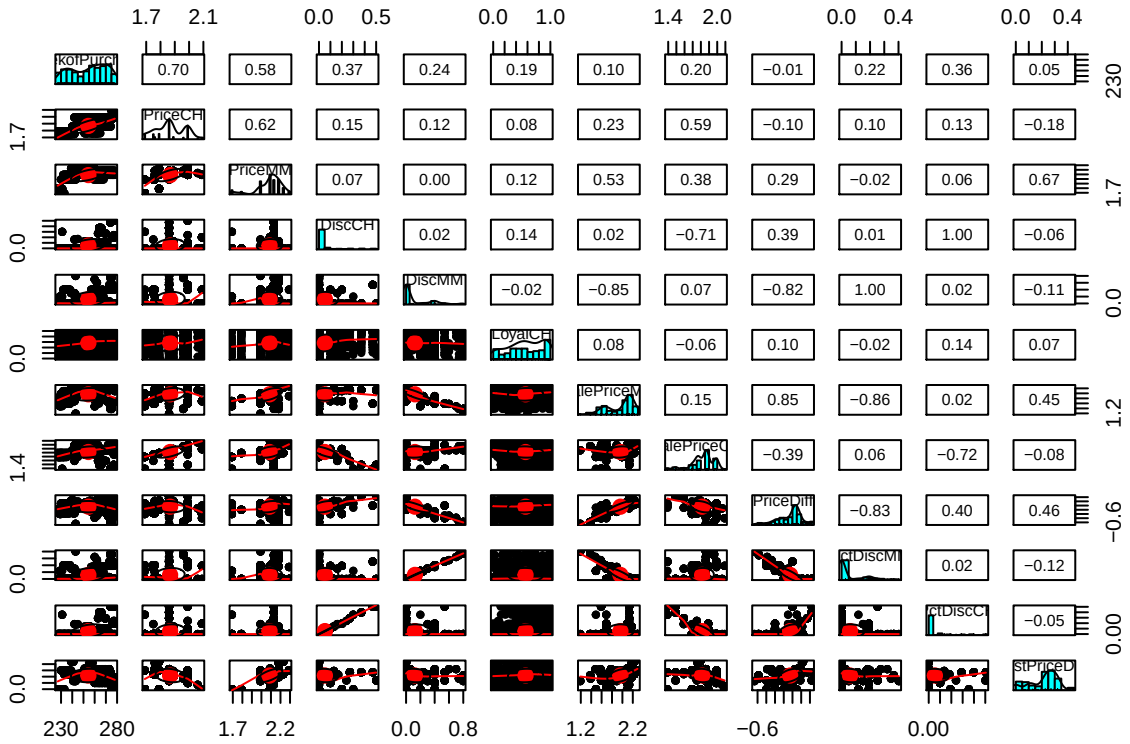
```
## Warning: package 'psych' was built under R version 3.5.2
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:kernlab':
##
##     alpha
```

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

```
pairs.panels(df[,unlist(lapply(df, is.numeric))])
```




```
library(PerformanceAnalytics)
```

```
## Warning: package 'PerformanceAnalytics' was built under R version 3.5.2
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

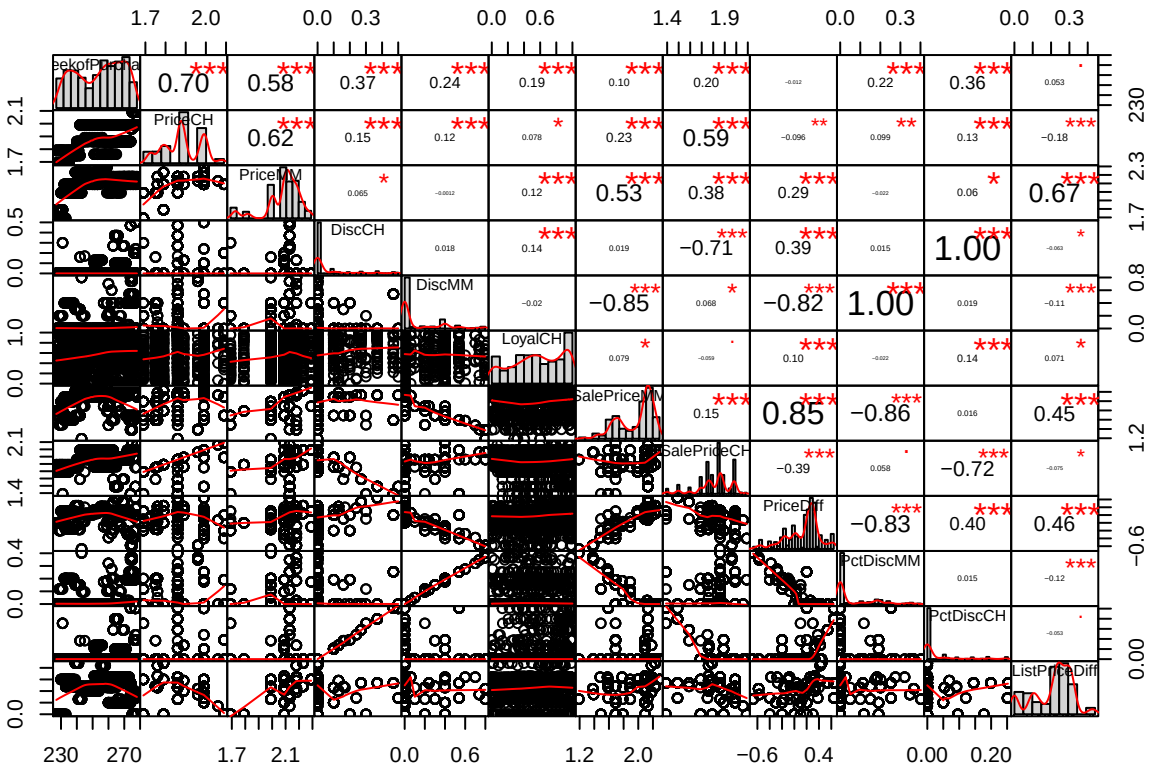
```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
##
## Attaching package: 'PerformanceAnalytics'
```

```
## The following object is masked from 'package:graphics':
##
##     legend
```

```
chart.Correlation(df[,unlist(lapply(df, is.numeric))])
```



Eliminating features based on correlation plot

The following are the list of set of features that have greater than or equal to |0.7| correlation:

WeekofPurchase PriceCH 0.70

DiscCH SalePriceCH -0.71

DiscCH PctDiscCH 1.00

DiscMM SalePriceMM -0.85

DiscMM PriceDiff -0.82

DiscMM PctDiscMM 1.00

SalePriceMM PriceDiff 0.85

SalePriceMM PctDiscMM -0.86

SalePriceCH PctDiscCH -0.72

PriceDiff PctDiscMM -0.83

Summarizing the correlated features:

PriceCH : WeekofPurchase

PctDiscMM : PriceDiff, SalePriceMM, DiscMM

PctDiscCH : SalePriceCH, DiscCH

Hence, we can eliminate the features WeekofPurchase, PriceDiff, SalePriceMM, DiscMM, SalePriceCH, DiscCH.

Redundant Features

```
library("dataPreparation")
```

```
## Warning: package 'dataPreparation' was built under R version 3.5.2
```

```
## Loading required package: lubridate
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loading required package: progress
```

```
## Warning: package 'progress' was built under R version 3.5.2
```

```
## dataPreparation 0.4.1
```

```
## Type dataPrepNews() to see new features/changes/bug fixes.
```

```
# IDENTIFY AND LIST VARIABLES THAT ARE CONSTANTS
constant_cols <- whichAreConstant(df)
```

```
## [1] "whichAreConstant: it took me 0.01s to identify 0 constant column(s)"
```

```
# IDENTIFY AND LIST VARIABLES THAT ARE DOUBLES
double_cols <- whichAreInDouble(df)
```

```
## [1] "whichAreInDouble: it took me 0.01s to identify 0 column(s) to drop."
```

```
# IDENTIFY AND LIST VARIABLES THAT ARE EXACT BIJECTIONS
bijections_cols <- whichAreBijection(df)
```

```
## [1] "whichAreBijection: STORE is a bijection of StoreID. I put it in drop list."
## [1] "whichAreBijection: it took me 0.09s to identify 1 column(s) to drop."
```

It shows that STORE and StoreID are same. One of them has to be removed. And Store7 information is there as part of StoreID. Hence, STORE and Store7 can be removed.

Remaining list of Predictors:

StoreID

PriceCH

PriceMM

SpecialCH

SpecialMM

LoyalCH

PctDiscMM

PctDiscCH

ListPriceDiff

Converting the values of CH and MM into 0 and 1 for final prediction

```
str(df$Purchase)
```

```
##  Factor w/ 2 levels "CH","MM": 1 1 1 2 1 1 1 1 1 1 ...
```

```
table(df$Purchase)
```

```
##
##  CH  MM
## 653 417
```

```
df$Purchase <- ifelse(df$Purchase=="MM",1,0)
table(df$Purchase)
```

```
##
##   0   1
## 653 417
```

```
str(df$Purchase)
```

```
##  num [1:1070] 0 0 0 1 0 0 0 0 0 0 ...
```

```
df$Purchase <- as.factor(df$Purchase)
str(df$Purchase)
```

```
##  Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 1 1 ...
```

```
table(df$Purchase)
```

```
##
##   0   1
## 653 417
```

Handling Overfitting

Splitting into train and test data

```
split = 0.7
set.seed(100)

train_index <- sample(1:nrow(df), split * nrow(df))
test_index <- setdiff(1:nrow(df), train_index)

X_train <- df[train_index,]
X_test <- df[test_index,]
```

For Cross Validation

```
control <- trainControl(method = "repeatedcv",
                        number = 10,
                        repeats = 4)
```

Caret package works good for handling all these different models. It takes care of cross validation for handling over-filling and scaling by pre-processing the data with 'center' and 'scale'.

Various Models

Logistic using caret package

```
logistic_model <- train(Purchase ~ StoreID +
                          PriceCH +
                          PriceMM +
                          SpecialCH +
                          SpecialMM +
                          LoyalCH +
                          PctDiscMM +
                          PctDiscCH +
                          ListPriceDiff,
                        data = X_train,
                        method = "glm",
                        preProcess = c("center", "scale"),
                        family = binomial(link = 'logit'),
                        trControl = control)

summary(logistic_model)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8234  -0.5360  -0.2144   0.5022   2.8868
##
## Coefficients: (1 not defined because of singularities)
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.86788    0.11718  -7.406  1.3e-13 ***
## StoreID2      -0.07863    0.13520  -0.582 0.560875
## StoreID3      -0.01777    0.16286  -0.109 0.913094
## StoreID4      -0.05659    0.14814  -0.382 0.702454
## StoreID7      -0.41304    0.16363  -2.524 0.011593 *
## PriceCH        0.33968    0.17391   1.953 0.050801 .
## PriceMM       -0.46850    0.13807  -3.393 0.000691 ***
## SpecialCH1    -0.01583    0.14774  -0.107 0.914683
## SpecialMM1     0.08938    0.12700   0.704 0.481564
## LoyalCH       -1.99882    0.15707 -12.726  < 2e-16 ***
## PctDiscMM      0.47597    0.13908   3.422 0.000621 ***
## PctDiscCH     -0.37276    0.15520  -2.402 0.016315 *
## ListPriceDiff       NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 999.87  on 748  degrees of freedom
```

```
## Residual deviance: 559.06  on 737  degrees of freedom
## AIC: 583.06
##
## Number of Fisher Scoring iterations: 5
```

```
logistic_model
```

```
## Generalized Linear Model
##
## 749 samples
##   9 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold, repeated 4 times)
## Summary of sample sizes: 674, 674, 675, 674, 674, 674, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8264099  0.6305658
```

```
#Prediction using logistic regression
#Binary outcome
X_test$logistic_prediction <- predict(logistic_model, newdata = X_test)
confusionMatrix(data = X_test$logistic_prediction, X_test$Purchase)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 177  35
##          1  17  92
##
##                Accuracy : 0.838
##                  95% CI : (0.7931, 0.8766)
##     No Information Rate : 0.6044
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6528
##  Mcnemar's Test P-Value : 0.0184
##
##             Sensitivity : 0.9124
##             Specificity : 0.7244
##          Pos Pred Value : 0.8349
##          Neg Pred Value : 0.8440
##              Prevalence : 0.6044
##          Detection Rate : 0.5514
##    Detection Prevalence : 0.6604
##       Balanced Accuracy : 0.8184
##
##        'Positive' Class : 0
##
```

The warning 'prediction from a rank-deficient fit may be misleading' is displayed in the above model. To ignore the warning message, 'message=FALSE, warning=FALSE' is added to the r chunk. On researching further, it appears that the warning message is due to the fact that estimate for ListPriceDiff is NA. On

analysing this further more, we see that PriceMM - PriceCH = ListPriceDiff. ListPriceDiff is redundant information and it does not add any value to the model. Hence this feature can be eliminated.

Final List of Predictors: To summarize, the features WeekofPurchase, PriceDiff, SalePriceMM, DiscMM, SalePriceCH, DiscCH are eliminated due to multi-collinearity in the data. The features STORE and StoreID represent one and the same. Store7 information is part of StoreID. Hence the features STORE and Store7 can be eliminated. On analyzing furthermore, we see that ListPriceDiff does not add any value to the model. Hence, that can be eliminated too. Now, the final list of predictors are:

StoreID

PriceCH

PriceMM

SpecialCH

SpecialMM

LoyalCH

PctDiscMM

PctDiscCH

Logistic without ListPriceDiff using caret package

```
logistic_model_nolistpricediff <- train(Purchase ~ StoreID +
                          PriceCH +
                          PriceMM +
                          SpecialCH +
                          SpecialMM +
                          LoyalCH +
                          PctDiscMM +
                          PctDiscCH,
                        data = X_train,
                        method = "glm",
                        preProcess = c("center", "scale"),
                        family = binomial(link = 'logit'),
                        trControl = control)

summary(logistic_model_nolistpricediff)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8234  -0.5360  -0.2144   0.5022   2.8868
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.86788    0.11718  -7.406  1.3e-13 ***
## StoreID2    -0.07863    0.13520  -0.582 0.560875
## StoreID3    -0.01777    0.16286  -0.109 0.913094
## StoreID4    -0.05659    0.14814  -0.382 0.702454
## StoreID7    -0.41304    0.16363  -2.524 0.011593 *
```

```
## PriceCH      0.33968    0.17391   1.953 0.050801 .
## PriceMM     -0.46850    0.13807  -3.393 0.000691 ***
## SpecialCH1  -0.01583    0.14774  -0.107 0.914683
## SpecialMM1   0.08938    0.12700   0.704 0.481564
## LoyalCH     -1.99882    0.15707 -12.726  < 2e-16 ***
## PctDiscMM    0.47597    0.13908   3.422 0.000621 ***
## PctDiscCH   -0.37276    0.15520  -2.402 0.016315 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 999.87  on 748  degrees of freedom
## Residual deviance: 559.06  on 737  degrees of freedom
## AIC: 583.06
##
## Number of Fisher Scoring iterations: 5
```

```
logistic_model_nolistpricediff
```

```
## Generalized Linear Model
##
## 749 samples
##   8 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold, repeated 4 times)
## Summary of sample sizes: 674, 674, 675, 674, 674, 674, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8234009  0.6236176
```

```
#Prediction using logistic regression without ListPriceDiff
#Binary outcome
X_test$logistic_nolistpricediff_prediction <- predict(logistic_model_nolistpricediff, newdata = X_test)
confusionMatrix(data = X_test$logistic_nolistpricediff_prediction , X_test$Purchase)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 177  35
##          1  17  92
##
##                Accuracy : 0.838
##                  95% CI : (0.7931, 0.8766)
##     No Information Rate : 0.6044
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6528
##  Mcnemar's Test P-Value : 0.0184
##
##             Sensitivity : 0.9124
##             Specificity : 0.7244
```

```
##          Pos Pred Value : 0.8349
##          Neg Pred Value : 0.8440
##              Prevalence : 0.6044
##          Detection Rate : 0.5514
##    Detection Prevalence : 0.6604
##       Balanced Accuracy : 0.8184
##
##        'Positive' Class : 0
##
```

Logistic Regression yields 83.8% accuracy. We see that both the logistic models give the same accuracy. Removing ListPriceDiff did not make any difference and removing this features is the right thing to do.

We see that PriceMM, LoyalCH and PctDiscMM are strongly significant predictors. The features PriceMM and LoyalCH have negative effect on purchasing the MM orange juice. PctDiscMM has positive effect on purchasing the MM orange juice. It totally makes sense. When the price of MM juice goes high, customers will tend to look for other brands of orange juice. Hence, for motivating the customers to buy MM orange juice, price of MM (PriceMM) should go down. Similarly, when the customers are more loyal to CH orange juice, they are less likely to buy MM orange juice. LoyalCH has negative effect on MM orange juice purchase. On the other hand, more discount on MM orange juice (PctDiscMM), boosts the sales of MM orange juice.

In addition to this, there is a slight negative effect on the purchase of MM orange juice when there is an increase on the features Store7 or PctDiscCH. It totally makes sense that the increase in discount for CH (PctDiscCH), tend the customers to buy CH, resulting in not buying MM orange juice. It looks like the customers who visit Store7 buy more of CH orange juice for whatsover be the reason. May be, they are more loyal customers to CH juice.

```
PriceMM_coef <- summary(logistic_model_nolistpricediff)$coefficients[7]
PriceMM_coef
```

```
## [1] -0.4685007
```

```
LoyalCH_coef <- summary(logistic_model_nolistpricediff)$coefficients[10]
LoyalCH_coef
```

```
## [1] -1.998817
```

```
PctDiscMM_coef <- summary(logistic_model_nolistpricediff)$coefficients[11]
PctDiscMM_coef
```

```
## [1] 0.4759681
```

```
#Converting into Probabilities
exp(PriceMM_coef) / (1 + exp(PriceMM_coef))
```

```
## [1] 0.3849712
```

```
exp(LoyalCH_coef) / (1 + exp(LoyalCH_coef))
```

```
## [1] 0.1193271
```

```
exp(PctDiscMM_coef) / (1 + exp(PctDiscMM_coef))
```

```
## [1] 0.6167954
```

On converting the log odds into probabilities, we see that PctDiscMM has the highest influence on purchasing MM orange juice.

Logistic Regression for probability detection

```
prob_log <- glm(Purchase ~ StoreID +
                           PriceCH +
                           PriceMM +
                           SpecialCH +
                           SpecialMM +
                           LoyalCH +
                           PctDiscMM +
                           PctDiscCH,
                         data = X_train,
                         family = binomial(link = 'logit'))

summary(prob_log)
```

```
##
## Call:
## glm(formula = Purchase ~ StoreID + PriceCH + PriceMM + SpecialCH +
##     SpecialMM + LoyalCH + PctDiscMM + PctDiscCH, family = binomial(link = "logit"),
##     data = X_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8234  -0.5360  -0.2144   0.5022   2.8868
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.80374    2.30313   1.652 0.098627 .
## StoreID2    -0.19084    0.32817  -0.582 0.560875
## StoreID3    -0.04484    0.41083  -0.109 0.913094
## StoreID4    -0.17829    0.46672  -0.382 0.702454
## StoreID7    -0.88262    0.34965  -2.524 0.011593 *
## PriceCH      3.38737    1.73430   1.953 0.050801 .
## PriceMM     -3.44171    1.01430  -3.393 0.000691 ***
## SpecialCH1  -0.04357    0.40666  -0.107 0.914683
## SpecialMM1   0.24189    0.34370   0.704 0.481564
## LoyalCH     -6.48260    0.50940 -12.726  < 2e-16 ***
## PctDiscMM    4.65442    1.36003   3.422 0.000621 ***
## PctDiscCH   -5.72320    2.38288  -2.402 0.016315 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 999.87  on 748  degrees of freedom
## Residual deviance: 559.06  on 737  degrees of freedom
## AIC: 583.06
##
## Number of Fisher Scoring iterations: 5
```

```
prob_log
```

```
##
## Call:  glm(formula = Purchase ~ StoreID + PriceCH + PriceMM + SpecialCH +
##     SpecialMM + LoyalCH + PctDiscMM + PctDiscCH, family = binomial(link = "logit"),
```

```
##     data = X_train)
##
## Coefficients:
## (Intercept)     StoreID2     StoreID3     StoreID4     StoreID7
##     3.80374     -0.19084     -0.04484     -0.17829     -0.88262
##     PriceCH      PriceMM   SpecialCH1   SpecialMM1      LoyalCH
##     3.38737     -3.44171     -0.04357      0.24189     -6.48260
##   PctDiscMM    PctDiscCH
##     4.65442     -5.72320
##
## Degrees of Freedom: 748 Total (i.e. Null);  737 Residual
## Null Deviance:       999.9
## Residual Deviance: 559.1     AIC: 583.1
```

```
#Probability outcome
X_test$logistic_prob_prediction <- predict(prob_log, newdata = X_test, type = "response")
```

SVM Linear using caret package

```
svmLinear_model <- train(Purchase ~ StoreID +
                           PriceCH +
                           PriceMM +
                           SpecialCH +
                           SpecialMM +
                           LoyalCH +
                           PctDiscMM +
                           PctDiscCH,
                         data = X_train,
                         method = "svmLinear",
                         preProcess = c("center", "scale"),
                         trControl = control)

summary(svmLinear_model)
```

```
## Length  Class   Mode
##      1   ksvm     S4
```

```
svmLinear_model
```

```
## Support Vector Machines with Linear Kernel
##
## 749 samples
##   8 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold, repeated 4 times)
## Summary of sample sizes: 674, 674, 674, 674, 674, 674, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8303874  0.6388165
##
## Tuning parameter 'C' was held constant at a value of 1
```

```r
#Prediction using SVM Linear
X_test$svmLinear_prediction <- predict(svmLinear_model, newdata = X_test)
confusionMatrix(data = X_test$svmLinear_prediction, X_test$Purchase)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 175  32
##          1  19  95
##
##                Accuracy : 0.8411
##                  95% CI : (0.7965, 0.8794)
##     No Information Rate : 0.6044
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.6618
##  Mcnemar's Test P-Value : 0.09289
##
##             Sensitivity : 0.9021
##             Specificity : 0.7480
##          Pos Pred Value : 0.8454
##          Neg Pred Value : 0.8333
##              Prevalence : 0.6044
##          Detection Rate : 0.5452
##    Detection Prevalence : 0.6449
##       Balanced Accuracy : 0.8250
##
##        'Positive' Class : 0
##
```

SVM Linear model yields 84.11% accuracy, which is slightly more than the logistic regression.

SVM Radial using caret package

```r
grid_radial <- expand.grid(sigma = c(0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9),
                           C = c(0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 1, 2))
svmRadial_model <- train(Purchase ~ StoreID +
                           PriceCH +
                           PriceMM +
                           SpecialCH +
                           SpecialMM +
                           LoyalCH +
                           PctDiscMM +
                           PctDiscCH,
                         data = X_train,
                         method = "svmRadial",
                         preProcess = c("center", "scale"),
                         tuneGrid = grid_radial,
                         trControl = control)

summary(svmRadial_model)
```

```
## Length  Class   Mode
```

```
##      1   ksvm     S4
```

```
svmRadial_model
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 749 samples
##   8 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold, repeated 4 times)
## Summary of sample sizes: 674, 674, 675, 674, 674, 674, ...
## Resampling results across tuning parameters:
##
##   sigma  C     Accuracy   Kappa
##   0.01   0.01  0.6128108   0.000000000
##   0.01   0.05  0.6128108   0.000000000
##   0.01   0.10  0.7973694   0.540056345
##   0.01   0.25  0.8214144   0.615712015
##   0.01   0.50  0.8207568   0.615869407
##   0.01   0.75  0.8254189   0.626203877
##   0.01   0.90  0.8257432   0.626826100
##   0.01   1.00  0.8267432   0.629032554
##   0.01   2.00  0.8277387   0.630999217
##   0.05   0.01  0.6128108   0.000000000
##   0.05   0.05  0.8087117   0.578561512
##   0.05   0.10  0.8197207   0.611915338
##   0.05   0.25  0.8247252   0.622701188
##   0.05   0.50  0.8243874   0.622006884
##   0.05   0.75  0.8260631   0.626423143
##   0.05   0.90  0.8250541   0.625039288
##   0.05   1.00  0.8240495   0.623054239
##   0.05   2.00  0.8190360   0.612962900
##   0.10   0.01  0.6128108   0.000000000
##   0.10   0.05  0.8067117   0.566293996
##   0.10   0.10  0.8120270   0.593267973
##   0.10   0.25  0.8163784   0.605378598
##   0.10   0.50  0.8223739   0.618982933
##   0.10   0.75  0.8223784   0.619524487
##   0.10   0.90  0.8207117   0.615922087
##   0.10   1.00  0.8197162   0.613960044
##   0.10   2.00  0.8163784   0.607310635
##   0.25   0.01  0.6128108   0.000000000
##   0.25   0.05  0.6775586   0.204557492
##   0.25   0.10  0.7900270   0.527228525
##   0.25   0.25  0.8110360   0.592204522
##   0.25   0.50  0.8160541   0.605211542
##   0.25   0.75  0.8140541   0.601481500
##   0.25   0.90  0.8117207   0.597197181
##   0.25   1.00  0.8093874   0.592364965
##   0.25   2.00  0.7990360   0.570741724
##   0.50   0.01  0.6128108   0.000000000
##   0.50   0.05  0.6451802   0.102817033
##   0.50   0.10  0.7219189   0.333913095
```

```
##   0.50   0.25  0.7926892   0.543787660
##   0.50   0.50  0.7986892   0.564240989
##   0.50   0.75  0.7970315   0.563351589
##   0.50   0.90  0.7940270   0.557790770
##   0.50   1.00  0.7916892   0.553734260
##   0.50   2.00  0.7866937   0.544524519
##   0.75   0.01  0.6128108   0.000000000
##   0.75   0.05  0.6154910   0.009144007
##   0.75   0.10  0.6982387   0.262040184
##   0.75   0.25  0.7846667   0.519060101
##   0.75   0.50  0.7933288   0.550484396
##   0.75   0.75  0.7890090   0.544905244
##   0.75   0.90  0.7876757   0.542925185
##   0.75   1.00  0.7840045   0.536298297
##   0.75   2.00  0.7766712   0.523617158
##   0.90   0.01  0.6128108   0.000000000
##   0.90   0.05  0.6121441  -0.001322993
##   0.90   0.10  0.6902387   0.238273872
##   0.90   0.25  0.7773153   0.499134444
##   0.90   0.50  0.7903198   0.542495061
##   0.90   0.75  0.7869955   0.540115039
##   0.90   0.90  0.7856712   0.538798469
##   0.90   1.00  0.7830090   0.534147649
##   0.90   2.00  0.7733423   0.516136936
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.01 and C = 2.
```

```
#Prediction using SVM Radial
X_test$svmRadial_prediction <- predict(svmRadial_model, newdata = X_test)
confusionMatrix(data = X_test$svmRadial_prediction, X_test$Purchase)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 177  35
##          1  17  92
##
##                Accuracy : 0.838
##                  95% CI : (0.7931, 0.8766)
##     No Information Rate : 0.6044
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6528
##  Mcnemar's Test P-Value : 0.0184
##
##             Sensitivity : 0.9124
##             Specificity : 0.7244
##          Pos Pred Value : 0.8349
##          Neg Pred Value : 0.8440
##              Prevalence : 0.6044
##          Detection Rate : 0.5514
##    Detection Prevalence : 0.6604
##       Balanced Accuracy : 0.8184
```

```
##
##        'Positive' Class : 0
##
```

SVM Radial model yields 83.8% accuracy, which is same as the accuracy of the logistic regression model, slightly lower than SVM Linear model.

Results and Conclusion

Brand Manager:

From the above analyses, we can conclude that the predictors PriceMM, LoyalCH and PctDiscMM influence the purchase of MM orange juice, The increase in PriceMM or LoyalCH will decrease the chances of buying MM orange juice. The increase in PctDiscMM will increase the chances of buying MM orange juice. The next important predictors are Store7 and PctDiscCH. For some reasons, the customers who buy at Store7 are more towards buying CH orange juice over MM orange juice. The increase in PctDiscCH decrease the chances of buying MM orange juice.

All the 3 models show more than 80% accuracy and gives good confidence on the models. Out of all the 3 models, SVM Linear has the highest accuracy level of 84.11%, which is slightly more than the logistic model and SVM Radial model which each have the accuracy level of 83.8%. If 'True Positive Rate' needs to be given the highest priority, then both logistic and SVM Radial has slightly higher Sensitivity of 91.24% when compare to SVM Linear with Sensitivity of 90.21%.

So, to summarize, one most important thing the customers are looking for is more discount on MM orange juice for purchasing MM orange juice.

SalesManager:

Logistic regression model gives the probability of customers buying MM orange juice. SVM Linear model is the good one with the highest accuracy of 84.11%. With such high accuracy, we can be confident with this model. If we are particular about Sensitivity, we can go with the SVM Radial model or Logistic model.

References

https://www.kaggle.com

https://stackoverflow.com

https://www.machinelearningplus.com/machine-learning/caret-package/