Machine Learning Project: Orange Juice Analysis

Siddharth Hatkar

11/18/2019

Contents

Problem:

The Branch Manager wants to know what variables or attributes are responsible for the customer's decision to buy MM. Also, he wants to know how we can improve the sales of MM. Here, the problem is that several variables can be correlated to the decision of a customer to buy MM or not. Like there may be some factor which, if increased, may increase the probability of the purchase or, if decreased, may lower the chances of the customer buying MM. Also, there may be statistically insignificant and does not influence the outcome variable. If we consider all these attributes for analysis, our model becomes more complex and thus more difficult to analyze (thus, not following Occam's razor). Thus, we need a simpler model for our analysis, and for that first, we need to decide which attributes to select.

On the other hand, the Sales Manager is more focused on predicting the chances of a customer buying MM, and he wants a predictive model that can do this task. The problem with the predictive model is that it will only provide a rate or probability of an outcome to happen, i.e., we still are not sure what the outcome will be, and we are only guessing logically. Also, to calculate the probability of a customer buying MM, we need to know the variables which are influencing the outcome. Then we need to choose a method to perform prediction on our model, and this method should have good accuracy(like low AIC).

Objective: - To find out varaibles which can influence the probabilty of a customer to purchase MM. - To create a model which can predict the probability of a customer to purchase MM.

Methods:

- Logistic Regression
- SVM
  - Linear
  - Radial

1 Loading Data and Libraries

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

```
## Loading required package: lubridate

##
## Attaching package: 'lubridate'

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

## Loading required package: stringr

## Loading required package: Matrix

## Loading required package: progress

## dataPreparation 0.4.1

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

```r
library("mlbench")
library("e1071")
library("caret")
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```r
library("ROCR")
```

```
## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

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

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

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

```
library("corrplot")
```

```
## corrplot 0.84 loaded
```

```
library("caret")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

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

2 Data Preparation

Before creating the model, we need to prepare our data in such a way that it will not hinder the analysis and model creation. This can be done using Exploratory Data Analysis techniques. ## 2.1 Data Cleaning: In this section, we are going to remove outliers, NAs and any unecessary attribute from our data frame. ### 2.1.1 Checking for outliers

```
summary(OJ)
```

```
##  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
```

2.1.2 Remove NAs

First, we need to prepare the data to apply Logistic Regression or SVM. To do this, we need to remove NA from our data, if any present.

```r
#check for NA
lapply(OJ, function(x) sum(is.na(x)))
```

```
## $Purchase
## [1] 0
##
## $WeekofPurchase
## [1] 0
##
## $StoreID
## [1] 0
##
## $PriceCH
## [1] 0
##
## $PriceMM
## [1] 0
##
## $DiscCH
## [1] 0
##
## $DiscMM
## [1] 0
##
## $SpecialCH
## [1] 0
##
## $SpecialMM
```

```
## [1] 0
##
## $LoyalCH
## [1] 0
##
## $SalePriceMM
## [1] 0
##
## $SalePriceCH
## [1] 0
##
## $PriceDiff
## [1] 0
##
## $Store7
## [1] 0
##
## $PctDiscMM
## [1] 0
##
## $PctDiscCH
## [1] 0
##
## $ListPriceDiff
## [1] 0
##
## $STORE
## [1] 0
```

2.1.3 Remove Unnecesaary variables

Now, we need to remove the varaibles from our data frame which are constant, double, bijection or included.

```
## Removing irrelevant variables

constant_cols <- whichAreConstant(OJ)
```

```
## [1] "whichAreConstant: it took me 0.02s to identify 0 constant column(s)"
```

```
double_cols <- whichAreInDouble(OJ)
```

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

```
bijections_cols <- whichAreBijection(OJ)
```

```
## [1] "whichAreBijection: STORE is a bijection of StoreID. I put it in drop list."
## [1] "whichAreBijection: it took me 0.14s to identify 1 column(s) to drop."
```

```
#The above results shows that STORE can be derived from StoreID. Thus, there is no need to consider STO
```

Removed the following variables from the dataframe:

- STORE

```r
#Store 7 and Store are redundant variables. These two varaibles provide same information as the StoreID
#STORE is a bijection of STOREID


New_OJ <- OJ[, c(-18)]
included_cols <- whichAreIncluded(New_OJ)
```

```
## [1] "whichAreIncluded: Store7 is included in column StoreID."
## [1] "whichAreIncluded: DiscCH is included in column PctDiscCH."
## [1] "whichAreIncluded: DiscMM is included in column PctDiscMM."
```

```r
New_OJ <- New_OJ[, c(-6,-7,-14)]
#As you can see the above results, there are some vaiables in our data frame that can be derived from o
```

- Store7
- DiscCH
- DiscMM

2.1.4 Factorizing Attributes

```r
#Finding the attributes which need to be factored
str(New_OJ)
```

```
## Classes 'data.table' and 'data.frame':   1070 obs. of  14 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 ...
##  $ 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 ...
##  $ 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 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
#factorizing the required attributes
New_OJ$StoreID <- as.factor(New_OJ$StoreID)
New_OJ$Purchase <- ifelse(New_OJ$Purchase == "CH", 1, 0)

str(New_OJ)
```

```
## Classes 'data.table' and 'data.frame':   1070 obs. of  14 variables:
##  $ Purchase      : num  1 1 1 0 1 1 1 1 1 1 ...
##  $ WeekofPurchase: int  237 239 245 227 228 230 232 234 235 238 ...
##  $ StoreID       : Factor w/ 5 levels "1","2","3","4",..: 1 1 1 1 5 5 5 5 5 5 ...
##  $ 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 ...
##  $ 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 ...
##  $ 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 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

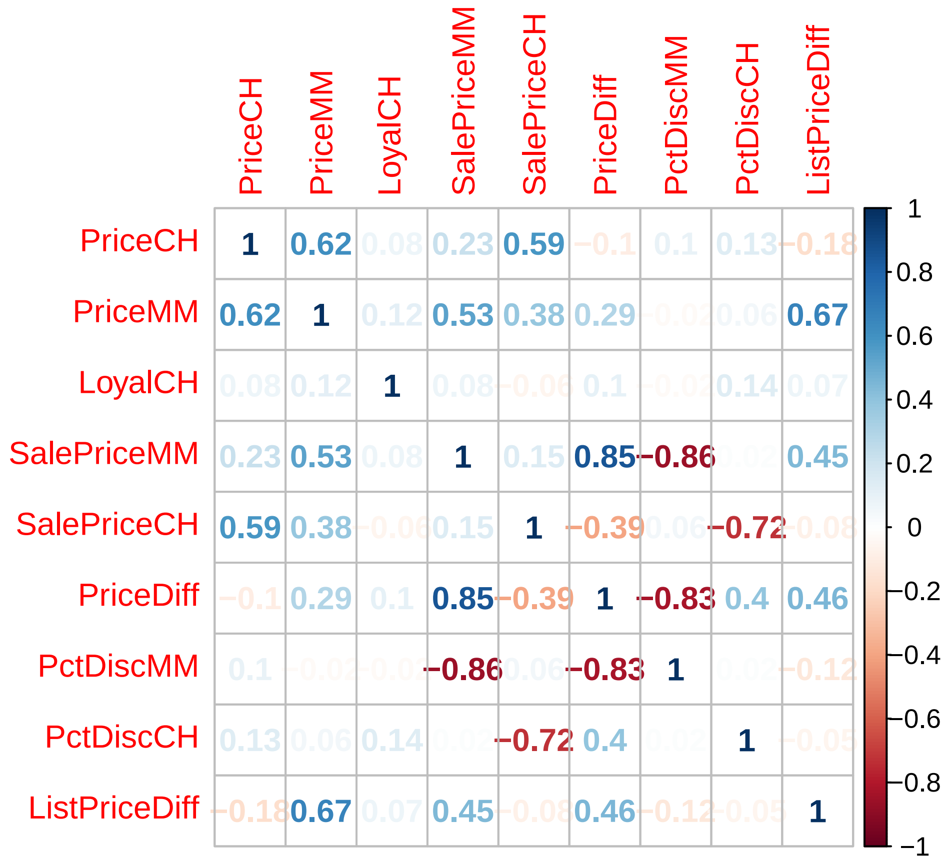
2.1.5 Removing Highly Correlated Variables

We remove highly correlated variable to elimate the redundancy in our model and decrease the complexity.

```
# Correlation
Cor_data <- New_OJ[, c("PriceCH", "PriceMM", "LoyalCH", "SalePriceMM", "SalePriceCH", "PriceDiff", "PctD
cor_result <- cor(Cor_data)
cor_result
```

```
##                  PriceCH     PriceMM     LoyalCH SalePriceMM SalePriceCH
## PriceCH       1.00000000  0.61640175  0.07779263  0.22938272  0.58671585
## PriceMM       0.61640175  1.00000000  0.11556956  0.53285867  0.38494127
## LoyalCH       0.07779263  0.11556956  1.00000000  0.07863126 -0.05888708
## SalePriceMM   0.22938272  0.53285867  0.07863126  1.00000000  0.14722240
## SalePriceCH   0.58671585  0.38494127 -0.05888708  0.14722240  1.00000000
## PriceDiff    -0.09633508  0.29259440  0.10426083  0.85279789 -0.39099950
## PctDiscMM     0.09915740 -0.02174741 -0.02246037 -0.85674903  0.05845905
## PctDiscCH     0.13460070  0.05996353  0.13868388  0.01621623 -0.72277560
## ListPriceDiff -0.17793470  0.66518696  0.07065930  0.44839527 -0.07529368
##                PriceDiff   PctDiscMM   PctDiscCH ListPriceDiff
## PriceCH       -0.09633508  0.09915740  0.13460070   -0.17793470
## PriceMM        0.29259440 -0.02174741  0.05996353    0.66518696
## 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
```

```
corrplot(cor_result, method="number")
```



As we can see in the correlation graph above, the variables with value closer to 1 or -1 have strong correlation. The sign indicate whether they are proportional or inversely proportional.

```
#Removing attributes having high correlation
#SalePriceCH
#SalePriceMM
#PriceDiff
#ListPriceDiff

New_OJ <- New_OJ[, c(-9, -10, -11, -14)]
```

After finding correlation, we have removed the following variables from the dataframe:

- SalePriceCH
- SalePriceMM
- PriceDiff
- ListPriceDiff

Till now, we have removed all the redundant and irrelevant variables form our dataframe. By removing unnecessary variables, we are increasing the accuracy and decreasing the complexity of our model. If we don't remove the unnecessary variables then it will not follow occam's law of parsimony and our model will become more hard to anaylze.

Afte removal of unnecessary variables, the variables that we are going to use in our model creations are:

- Purchase
- WeekOfPurchase
- StoreID
- PriceCH
- PriceMM
- SpecialCH
- SpecialMM
- LoyalCH
- PctDiscMM
- PctDiscCH

2 Reducing Overfitting

2.1 Splitting the data into Test and Train For Logistic Regression

To know whether our model is able to predict correct outcomes or not. We generally create our model using train data and for validation and testing, we use the test data. Train and test data, both are part of data frame. Here, we are using this technique to split data frame into Train and Test in the ration of 4:1 (i.e., value of split is 0.8) for Logistic Regression.

2.2 Using Cross Validation for SVM

Using 4-fold Cross validation for SVM to optimize the training set and testing set and thus reducing over-fitting.

```r
split = 0.8
set.seed(99894)

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

OJ_train <- New_OJ[train_index,]
OJ_test <- New_OJ[test_index,]
OJ_train
```

```
##      Purchase WeekofPurchase StoreID PriceCH PriceMM SpecialCH SpecialMM
##   1:        0            271       3    1.99    2.09         1         0
##   2:        1            237       2    1.75    1.99         0         0
##   3:        1            269       7    1.86    2.13         1         0
##   4:        0            256       7    1.86    2.18         0         0
##   5:        0            236       1    1.75    1.99         0         0
##  ---
## 852:        1            229       2    1.69    1.69         0         0
## 853:        0            275       1    1.96    2.13         0         1
## 854:        0            270       3    1.99    2.09         0         0
## 855:        1            258       4    1.99    2.29         0         0
## 856:        0            232       2    1.69    1.69         1         0
##       LoyalCH PctDiscMM PctDiscCH
##   1: 0.500000  0.191388  0.050251
##   2: 0.484608  0.000000  0.000000
##   3: 0.754240  0.000000  0.145161
##   4: 0.131072  0.000000  0.000000
```

```
##   5: 0.548160  0.000000  0.000000
##  ---
## 852: 0.795200  0.000000  0.000000
## 853: 0.863685  0.347418  0.000000
## 854: 0.000043  0.000000  0.050251
## 855: 0.640368  0.000000  0.000000
## 856: 0.680000  0.000000  0.000000
```

```
OJ_test
```

```
##      Purchase WeekofPurchase StoreID PriceCH PriceMM SpecialCH SpecialMM
##   1:        1            245       1    1.86    2.09         0         0
##   2:        0            227       1    1.69    1.69         0         0
##   3:        0            269       2    1.86    2.18         0         0
##   4:        1            259       7    1.86    2.18         0         0
##   5:        1            274       7    1.86    2.13         1         0
##  ---
## 210:        1            230       7    1.69    1.99         0         1
## 211:        1            237       7    1.75    1.99         0         0
## 212:        0            236       1    1.75    1.99         0         0
## 213:        1            252       7    1.86    2.09         0         0
## 214:        1            261       7    1.86    2.13         0         0
##       LoyalCH PctDiscMM PctDiscCH
##   1: 0.680000  0.000000  0.091398
##   2: 0.400000  0.000000  0.000000
##   3: 0.320000  0.000000  0.000000
##   4: 0.744000  0.000000  0.000000
##   5: 0.932891  0.253521  0.252688
##  ---
## 210: 0.595200  0.000000  0.000000
## 211: 0.740928  0.201005  0.000000
## 212: 0.695258  0.000000  0.000000
## 213: 0.587822  0.000000  0.053763
## 214: 0.588965  0.112676  0.000000
```

2 Logistic Regression

Using glm funtion on the train dataframe to perform logistic Regression on the binomial family since the Purchase, the outcome variable, has only two possible outcomes. Then using test dataframe to predict from our trained model. Also, analyzing the confusion matrix to find out the accuracy of the model and AIC value.

```
#model1
LR_clean_OJ <- glm(family = "binomial", Purchase ~., data = OJ_train)
summary(LR_clean_OJ)$coefficients
```

```
##                    Estimate Std. Error    z value     Pr(>|z|)
## (Intercept)    -1.909924991 2.04046370 -0.9360250 3.492603e-01
## WeekofPurchase  0.008369773 0.01240127  0.6749124 4.997314e-01
## StoreID2       -0.193421893 0.30351801 -0.6372666 5.239512e-01
## StoreID3        0.233574053 0.43422566  0.5379094 5.906396e-01
```

```
## StoreID4        0.647920555 0.46387547  1.3967554 1.624872e-01
## StoreID7        0.723586669 0.32149267  2.2507097 2.440393e-02
## PriceCH        -5.080755644 2.10801858 -2.4102044 1.594359e-02
## PriceMM         3.126551334 0.98842984  3.1631495 1.560722e-03
## SpecialCH      -0.178887494 0.36443643 -0.4908606 6.235250e-01
## SpecialMM      -0.135887120 0.30704011 -0.4425712 6.580759e-01
## LoyalCH         6.029411254 0.45059935 13.3808697 7.822663e-41
## PctDiscMM      -4.339956936 1.24046257 -3.4986601 4.676022e-04
## PctDiscCH       6.528185664 2.37715478  2.7462182 6.028666e-03
```

```r
LR_clean_OJ
```

```
##
## Call:  glm(formula = Purchase ~ ., family = "binomial", data = OJ_train)
##
## Coefficients:
##    (Intercept)  WeekofPurchase        StoreID2        StoreID3
##       -1.90992         0.00837        -0.19342         0.23357
##       StoreID4        StoreID7         PriceCH         PriceMM
##        0.64792         0.72359        -5.08076         3.12655
##      SpecialCH       SpecialMM         LoyalCH       PctDiscMM
##       -0.17889        -0.13589         6.02941        -4.33996
##      PctDiscCH
##        6.52819
##
## Degrees of Freedom: 855 Total (i.e. Null);  843 Residual
## Null Deviance:       1136
## Residual Deviance: 665.1     AIC: 691.1
```

```r
prediction <- predict(LR_clean_OJ, OJ_test, type = "response")
result <- ifelse(prediction > 0.50, '1', '0')

confusionMatrix(data = as.factor(result), as.factor(OJ_test$Purchase))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  69  11
##          1  24 110
##
##                Accuracy : 0.8364
##                  95% CI : (0.78, 0.8834)
##     No Information Rate : 0.5654
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.6617
##
##  Mcnemar's Test P-Value : 0.04252
##
##             Sensitivity : 0.7419
##             Specificity : 0.9091
##          Pos Pred Value : 0.8625
```

```
##          Neg Pred Value : 0.8209
##              Prevalence : 0.4346
##          Detection Rate : 0.3224
##    Detection Prevalence : 0.3738
##       Balanced Accuracy : 0.8255
##
##        'Positive' Class : 0
##
```

The P-values for SpecialCH, SpecialMM, and WeekOfPurchase are more than 0.05, which states that these variables will not influence the outcome varaible, i.e., Purchase. Therefore, we will remove these varaibles from our model.

Now, we will create a new model (model 2) with only the significant variables.

```r
LR_reduced_OJ <- glm(family = "binomial", Purchase ~ StoreID + PriceCH + PriceMM + LoyalCH + PctDiscMM +
summary(LR_reduced_OJ)$coefficients
```

```
##                 Estimate Std. Error    z value     Pr(>|z|)
## (Intercept) -2.16424344  1.9751455 -1.0957387 2.731931e-01
## StoreID2    -0.21469509  0.3014906 -0.7121120 4.763954e-01
## StoreID3     0.09781079  0.3681528  0.2656798 7.904858e-01
## StoreID4     0.52523350  0.4179193  1.2567820 2.088326e-01
## StoreID7     0.66938234  0.3003387  2.2287582 2.583000e-02
## PriceCH     -3.96457795  1.5125501 -2.6211217 8.764096e-03
## PriceMM      3.27662305  0.9517321  3.4427997 5.757256e-04
## LoyalCH      6.04572721  0.4469998 13.5251222 1.111589e-41
## PctDiscMM   -4.59954048  1.0087167 -4.5597942 5.120378e-06
## PctDiscCH    6.55270929  2.0538506  3.1904509 1.420510e-03
```

```r
LR_reduced_OJ
```

```
##
## Call:  glm(formula = Purchase ~ StoreID + PriceCH + PriceMM + LoyalCH +
##     PctDiscMM + PctDiscCH, family = "binomial", data = OJ_train)
##
## Coefficients:
## (Intercept)     StoreID2     StoreID3     StoreID4     StoreID7
##    -2.16424     -0.21470      0.09781      0.52523      0.66938
##     PriceCH      PriceMM      LoyalCH    PctDiscMM    PctDiscCH
##    -3.96458      3.27662      6.04573     -4.59954      6.55271
##
## Degrees of Freedom: 855 Total (i.e. Null);  846 Residual
## Null Deviance:       1136
## Residual Deviance: 665.9     AIC: 685.9
```

```r
prediction2 <- predict(LR_reduced_OJ, OJ_test, type = "response")
result2 <- ifelse(prediction > 0.50, 1, 0)

confusionMatrix(data = as.factor(result2), as.factor(OJ_test$Purchase))
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   0   1
##          0  69  11
##          1  24 110
##
##                Accuracy : 0.8364
##                  95% CI : (0.78, 0.8834)
##     No Information Rate : 0.5654
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.6617
##
##  Mcnemar's Test P-Value : 0.04252
##
##             Sensitivity : 0.7419
##             Specificity : 0.9091
##          Pos Pred Value : 0.8625
##          Neg Pred Value : 0.8209
##              Prevalence : 0.4346
##          Detection Rate : 0.3224
##    Detection Prevalence : 0.3738
##       Balanced Accuracy : 0.8255
##
##        'Positive' Class : 0
##
```

The AIC values show that the model with reduced variables (model 2) is better than the old model (model 1).

3 SVM

3.1 Preparing training and test data by using 4-fold Cross Validation

```r
#SVM

New_OJ2 <- OJ %>%
  mutate(
    Purchase = recode_factor(Purchase, "MM" = 'Y' , "CH" = 'N'),
    Purchase = factor(Purchase),
    StoreID = factor(StoreID),
    SpecialMM = factor(SpecialMM),
    SpecialCH = factor(SpecialCH)
    )


# IDENTIFYING VARIABLES THAT ARE EITHER CONSTANTS, DOUBLES or BIJECTIONS
# AND THEN ELIMINATING
b_vars <- whichAreBijection(New_OJ2)
```

```
## [1] "whichAreBijection: STORE is a bijection of StoreID. I put it in drop list."
```

```
## [1] "whichAreBijection: it took me 0.14s to identify 1 column(s) to drop."
```

```r
c_vars <- whichAreConstant(New_OJ2)
```

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

```r
d_vars <- whichAreInDouble(New_OJ2)
```

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

```r
b_vars
```

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

```r
c_vars
```

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

```r
d_vars
```

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

```r
New_OJ2 <- New_OJ2[,c(-18)]

# Removing Included Variables
i_vars <- whichAreIncluded(New_OJ2)
```

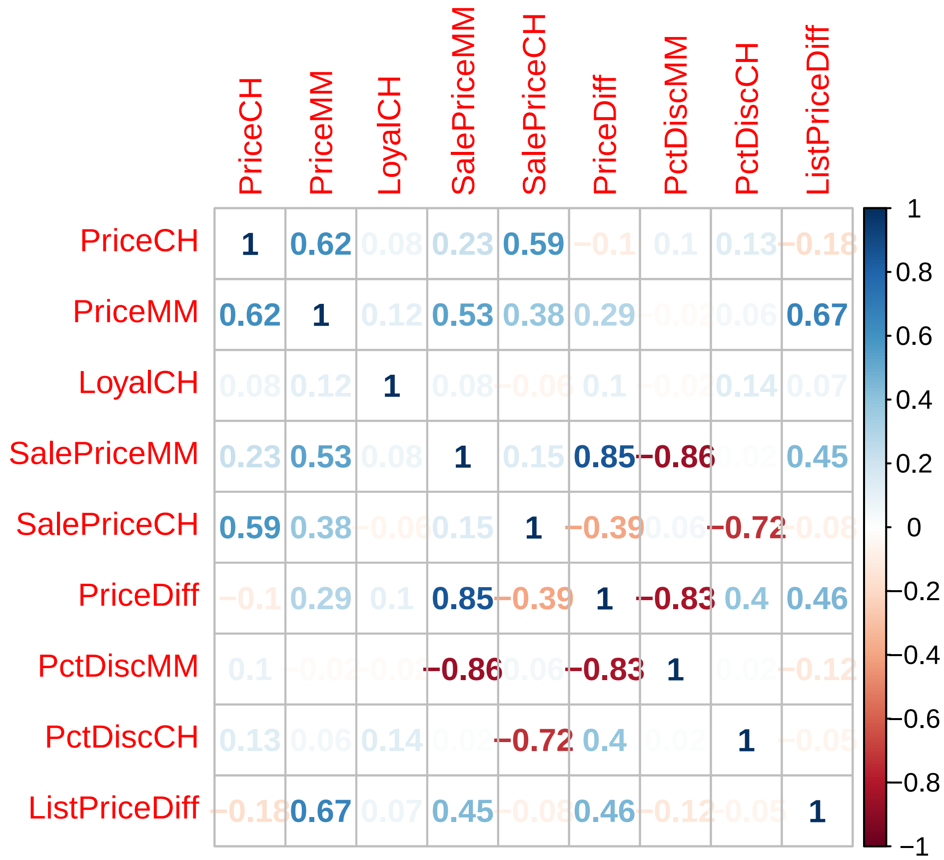
```
## [1] "whichAreIncluded: Store7 is included in column StoreID."
## [1] "whichAreIncluded: DiscCH is included in column PctDiscCH."
## [1] "whichAreIncluded: DiscMM is included in column PctDiscMM."
```

```r
i_vars
```

```
## [1]  6  7 14
```

```r
New_OJ2 <- New_OJ2[,c(-14,-7,-6)]

#Finding Correlation Matrix using numrical attributes
cor_data <- New_OJ2[, c(-1,-2,-3,-6,-7)]
corr_mat <- cor(cor_data)
corrplot(corr_mat, method = "number")
```




```
#Removing the highly correlated attributes

New_OJ2 <- New_OJ2[,c(-14,-11,-10,-9)]

#Model1 & Model2

New_OJ2 <- New_OJ2[,c(-2)]

#Code from Prof. Himanshu Mishra's SVM class

X_train_unscaled <- New_OJ2[train_index,-1]
y_train <- New_OJ2[train_index, 1]

X_test_unscaled <- New_OJ2[test_index, -1]
y_test <- New_OJ2[test_index, 1]


# DATA IS STANDARDIZED AND ENCODED (see see https://cran.r-project.org/web/packages/dataPreparation/vig
# Standardize continuous variables...
scales <- build_scales(dataSet = X_train_unscaled, cols = "auto", verbose = FALSE)

X_train <- fastScale(dataSet = X_train_unscaled, scales = scales, verbose = FALSE)
X_test <- fastScale(dataSet = X_test_unscaled, scales = scales, verbose = FALSE)

# Encode categorical variables...
encoding <- build_encoding(dataSet = X_train, cols = "auto", verbose = FALSE)
```

```r
X_train <- one_hot_encoder(dataSet = X_train, encoding = encoding, drop = TRUE, verbose = FALSE)
X_test <- one_hot_encoder(dataSet = X_test, encoding = encoding, drop = TRUE, verbose = FALSE)

# Create one data frame using both Outcome and Predictor Variables

train_Data <- cbind(y_train,X_train)
```

3.2 Implementing Radial SVM

Here, we are going to use radial SVM with different values of hyperparameters, i.e., C and Sigma. After this, using the trained model on the test data, and generating confusion matrix to get the accuracy of the model.

```r
fitControl <- trainControl(## 4-fold CV
  method = "repeatedcv",
  number = 4,
  ## repeated two times
  repeats = 2,
  summaryFunction=twoClassSummary,
  classProbs = TRUE)

grid <- expand.grid(sigma = c(.01,.05),
                    C = c(.05,.75,1,1.5,2))

# FIND OPTIMAL TUNING PARAMETERS (C and SIGMA)

svmFit1 <- train(Purchase ~ ., data = train_Data,
                 method='svmRadial',
                 trControl = fitControl,
                 metric = "ROC",
                 verbose = FALSE,
                 probability = TRUE,
                 tuneGrid = grid

)
#final values of hyperparameters; sigma = 0.01, C = 2

##Create a plot of ROC with with different values of C and gamma


svmFit1
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 856 samples
##  14 predictor
##   2 classes: 'Y', 'N'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold, repeated 2 times)
## Summary of sample sizes: 642, 642, 642, 642, 642, 642, ...
## Resampling results across tuning parameters:
```

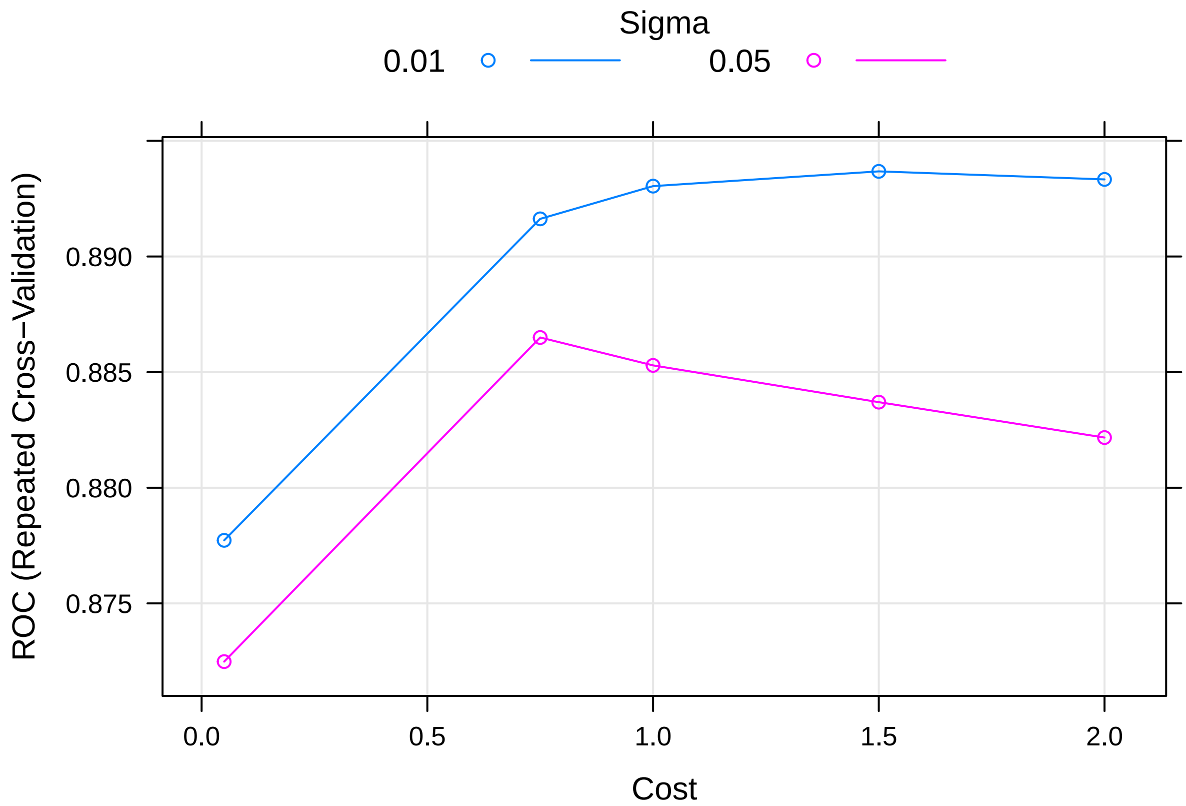
```
##
##   sigma  C     ROC        Sens       Spec
##   0.01   0.05  0.8777267  0.8441358  0.7612782
##   0.01   0.75  0.8916272  0.7253086  0.8778195
##   0.01   1.00  0.8930428  0.7314815  0.8834586
##   0.01   1.50  0.8936810  0.7391975  0.8796992
##   0.01   2.00  0.8933329  0.7268519  0.8787594
##   0.05   0.05  0.8724821  0.8101852  0.7781955
##   0.05   0.75  0.8864987  0.7021605  0.8796992
##   0.05   1.00  0.8852919  0.7098765  0.8778195
##   0.05   1.50  0.8837023  0.7037037  0.8787594
##   0.05   2.00  0.8821707  0.6975309  0.8825188
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.01 and C = 1.5.
```

```r
## Predict
svmPred <- predict(svmFit1, newdata = X_test, probability = TRUE)

confusionMatrix(data = svmPred, as.factor(y_test$Purchase))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Y   N
##          Y  70  11
##          N  23 110
##
##                Accuracy : 0.8411
##                  95% CI : (0.7851, 0.8874)
##     No Information Rate : 0.5654
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.6718
##
##  Mcnemar's Test P-Value : 0.05923
##
##             Sensitivity : 0.7527
##             Specificity : 0.9091
##          Pos Pred Value : 0.8642
##          Neg Pred Value : 0.8271
##              Prevalence : 0.4346
##          Detection Rate : 0.3271
##    Detection Prevalence : 0.3785
##       Balanced Accuracy : 0.8309
##
##        'Positive' Class : Y
##
```

```r
plot(svmFit1)
```



The above graph shows that the Sigma with value 0.01 is better than sigma with value 0.05. Since the area under the blue curve is more than that of red curve.

3.3 Implementing Linear SVM

Using Linear SVM with different values of hyperparameters, i.e., C and Sigma. After this, using the trained model on the test data, and generating confusion matrix to get the accuracy of the model.

```
#LinearSVM
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
svmFitL <- train(Purchase ~ ., data = train_Data,
                 method='svmLinear',
                 trControl = trctrl,
                 preProcess = c("center", "scale"),
                 tuneLength = 10)



svmFitL
```

```
## Support Vector Machines with Linear Kernel
##
## 856 samples
##  14 predictor
##   2 classes: 'Y', 'N'
##
```

```
## Pre-processing: centered (14), scaled (14)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 770, 769, 771, 771, 771, 770, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8254562  0.6236556
##
## Tuning parameter 'C' was held constant at a value of 1
```

```r
## Predict
svmPredL <- predict(svmFitL, newdata = X_test, probability = TRUE)

confusionMatrix(data = svmPredL, as.factor(y_test$Purchase))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Y   N
##          Y  72  12
##          N  21 109
##
##                Accuracy : 0.8458
##                  95% CI : (0.7903, 0.8914)
##     No Information Rate : 0.5654
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6827
##
##  Mcnemar's Test P-Value : 0.1637
##
##             Sensitivity : 0.7742
##             Specificity : 0.9008
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.8385
##              Prevalence : 0.4346
##          Detection Rate : 0.3364
##    Detection Prevalence : 0.3925
##       Balanced Accuracy : 0.8375
##
##        'Positive' Class : Y
##
```

4 Result and Conclusion

After analyzing both the supervised models, i.e., SVM and Logistic Regression, we came to the conclusion that the Linear SVM model is a better option. Linear SVM has given slightly beter accuracy than the other two models, which are Radial SVM and Logistic model.

4.1 Branch Manager's Questions

4.1.1 What predictor variables influence the purchase of MM?

- StoreID : StoreID
- LoyalCH : Customer brand loyalty for CH. That is, probability to buy CH (over MM) based on prior purchase behavior
- PriceCH :Price charged for CH. Also called List Price for CH
- PriceMM :Price charged for MM. Also called List Price for MM
- PctDiscCH :Percentage discount for CH
- PctDiscMM :Percentage discount for MM

4.1.2 Are all the variables in the dataset effective or are some more effective than others?

There are some variables which are statistically signicant and are more effective than others. These variables are as follows:

- LoyalCH
- PriceMM
- PctDiscMM
- PctDicCH

4.1.3 How confident are you in your recommendations?

In Logistic Regression table, it is clear that there are some variables with p value < 0.05. Thus, these variables have a significant impact on the outcome variable, i.e., Purchase. Also, the accuracy came 83.64%.

4.2 Sales Manager Questions

4.2.1 Can you provide him a predictive model that can tell him the probability of customers buying MM?

Logistic Regression is a better option since it does not make absoulte probability of a customer buying MM while SVM makes absolute prediction. Also, in LR, we can set a threshold to make the prediction. Also, the complexity of Logistic model is less compared to the SVM models.

4.2.2 How good is the model in its predictions?

After analyzing our model, we can say that our model is 83.64% accurate.

4.2.3 How confident are you in your recommendations?

With 95% Confidence Interval, our model covers the range from 0.78 to 0.8834 probability. Also, the p-value of our model is less than 0.05 which shows that our model is statistically significant.

5 Recommendation

To increase customers probability of buying MM, we can:

- increase the Discount on MM or reduce the price of MM or increase the price of CH
- Provide Loyalty points for MM same as CH.

6 Reference

- Lecture Handouts from Machine Learning class. Author: Prof. Himanshu Mishra
- https://www.guru99.com/r-apply-sapply-tapply.html
- https://stats.stackexchange.com/questions/95340/comparing-svm-and-logistic-regression
- https://rdrr.io/cran/ISLR/man/OJ.html
- https://cran.r-project.org/web/packages/ISLR/ISLR.pdf
- https://towardsdatascience.com/support-vector-machine-vs-logistic-regression-94cc2975433f