MKTG 6620: MachineLearning - Final Project

Willem van der Schans

11/20/2020

Contents

Business Problem

The client of this project is an unnamed grocery store chain. The client proposed questions regarding orange juice sales within the grocery store chain. As it stands two brands of orange juice are sold in the stores connected to the grocery store chain, namely Minute Maid and Citrus Hill. The proposed questions come from both the brand manager and the sales manager of the client and will be separately addressed within this report. Out of the two previously mentioned brands, Minute Maid is the brand that provides the stores with higher margin and is therefore the preferred brand to sell by the brand manager. To be able to direct efforts made by the client to sell Minute Maid over Citrus Hill, the brand manager is looking for insight into what factors influence Minute Maid sales over Citrus Hill sales. Next to the brand managers inquiry, the sales manager is looking for a predictive model to forecast sales of both brands. While no explicit motivation is given for the sales manager's inquiry, it can be assumed that forecasting sales will help the sales manager in his/her job of performance evaluation. This report contains code to reproduce the outcomes found throughout the project and to be able to apply the analysis on future data.

To summarize, this report sets out to firstly, find the reasons for consumers choosing for Minute Maid over Citrus Hill, and secondly, to create an as accurately as possible orange juice sales forecasting model. For both sections, specific recommendations are expected. More specifically, below are the questions to be answered in this report separated by which manager proposed the question.

Brand Manager

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 the analysis what are the specific recommendations?

Sales Manager

1. Provide 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?

To be able to answer all the questions above three machine learning algorithms are employed, namely Logistic-Regression, Step-Wise Logistic Regression and Support Vector Machine. Only the Support Vector Machine is used for forecasting sales, the other two mentioned models are used to give insights into consumer behavior.

Method

For this section of the report, the made choices will be explained alongside the corresponding code. This allows for grouping of code, output, and explanation together and prevents excessive referencing through the document.

Algorithms

Algorithms are picked based on the data, target variable, and the goal of the project. The first goal being variable selection and the fact that the target variable, categorically a logistic regression, is the optimal choice. The whitebox approach of logistic regression allows us to find which variables are most important for prediction and therefore have the biggest influence on customer behavior. After running a logistic regression model, a Step-Wise Logistic Regression will be run to perform Subset Selection. According to (Hastie, Friedman, & Tisbshirani, 2017), subset selection is beneficial because of firstly, better prediction accuracy (ties in to Sales Manager inquiries) by sacrificing a little bit of bias to reduce variance of the predicted values. Secondly, the interpretation becomes easier for the client. While this smaller data set is not much of a problem, it is still easier to interpret the Step-wise logistic regression results than the regular logistic regression results.

The second goal of this project is prediction with a categorial target variable. From experience, the most basic choices to do so are Naive Bayes, KNN, Random Forest, Logistic Regression, and SVM. When running tests on the dataset for this project, it turned out that SVM consistently performed the best prediction-wise which made it the final choice to use for this project. Since SVM is a black-box method it will not be able to provide any additional information towards the first goal.

```
# Base Libraries
library(tidyverse)
library(dplyr) # Additional Base level functionality of R
library(ggplot2) # Comprehensive Graphical Output Package
library(ggthemes) # More themes for GGPLOT2
library(Rfast) # Fast R Functions
library(bestNormalize) # Automatic Optimal Normalization
library(rlist) #list.append
library(scales) # Control over Axis formatting
library(kableExtra) # Output HTML Formatted Tables
library(reshape2) # Melt Function Correlation Matrix
library(lsr) #Correlation Matrix
library(stats) # Basic Statistical Functions
library(rlang) # Prepped Function <- Depreciates in 4.0.0
library(tictoc) # Timing of Code
library(lubridate) # Working with dates
library(gridExtra) # Plot in Grids
library(psych) #Describe Function
library(xtable)

# Machine Learning Libraries
library(e1071) #Skewness and other depreciated functions
library(caret) # MachineLearning package
library(doSNOW) # Parralel processing
library(rminer) #Mmetric Function
library(MASS) # Stepwise LM
library(randomForest) # RandomForest
```

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

factor_list <- c(names(df)[c(1,3,8,9,14,18)])

for (i in 1:length(factor_list)) {
  df[,factor_list[i]] <- as.factor(df[,factor_list[i]])
}

numeric_list <- c(names(df)[c(2,4,5,6,7,10,11,12,13,15,16,17)])

for (i in 1:length(numeric_list)) {
  df[,numeric_list[i]] <- as.numeric(df[,numeric_list[i]])
}

rm(factor_list, numeric_list)
```

Preliminary Analysis

The data supplied contains 1070 purchased of either Minute Maid or Citrus Hill over a time period of one year starting in week 1. A codebook of the data can be found in appendix A. A first look at the data provided by the summary function below, appendix B, shows no NA values or abnormal min or max values. The target variable of the data set is Purchase, which records "CH" for a Citrus Hill sale and "MM" for a Minute Maid sale. The distribution of the target variable is slightly skewed towards CH with 653 over 417. The slight skewness of the target variable means that the majority class predictor is 61.03% which will be the accuracy score to beat for the forecasting algorithm, the calculation can be found below.

```
paste0("The majority Class predictor benchmark is ",
       round((summary(df$Purchase)[1]/
                (summary(df$Purchase)[1]+summary(df$Purchase)[2])*100),2),"%")
```

```
## [1] "The majority Class predictor benchmark is 61.03%"
```

Initially, a lot of variables seem to hold overlapping information which will need to be further examined using a correlation matrix.

Multicollinearity

As mentioned above, the dataset seems to hold overlapping information mostly in its price related variables. When combing a number of variables, other variables can be derived. For example, ListPriceDiff is just PriceMM-PriceCH. Variables that hold the same information, or can be derived from each other, show strong correlation which creates Multicollinearity. Multicollinearity is an issue in regression analyses due to its negative effects on the model outcome. Firstly, Multicollinearity can make it hard to generalize or interpret model outcomes due to extreme sensitive coefficients, meaning the model outcome is useless in practice. Secondly, Multicollinearity can misrepresent the p-value making it hard to properly reject or retain the Null-Hypothesis. Both of these causes would interfere with the confidence of the recommendations made for the brand manager of the client. Therefore, Multicollinearity needs to be eliminated. To eliminate Multicollinearity we will follow the guidelines of correlation below and will eliminate any strong relationship of an absolute r above 0.7. (Mindrila & Balentyne).

Absolute Value of r Strength of Relationship

$r < 0.3$ None or very weak

$0.3 < r < 0.5$ Weak

$0.5 < r < 0.7$ Moderate

$r > 0.7$ Strong

Correlation Matrix Pre-Engineering

```
nums <- unlist(lapply(df, is.numeric))
Integers <- df[ , nums]

melted_cormat <- melt(cor(Integers))

names(melted_cormat)[3] <- "Correlation"

melted_cormat <- melted_cormat %>%
```

```
    arrange(Var1) %>%
    group_by(Var1) %>%
    filter(row_number() >= which(Var2 == Var1))

ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=Correlation)) +
  geom_tile(aes(fill=Correlation), col = "Black") +
  theme_fivethirtyeight () +
  theme(legend.position="right",legend.direction = "vertical",
        legend.text = element_text(size=7)) +
  geom_text(label = round(melted_cormat$Correlation, digits = 2),
            color="Black", size=3, alpha=0.5) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust=1, size=10),
        axis.text.y = element_text(angle=0,vjust = 0.5, hjust=1, size=10)) +
  scale_fill_gradient2_tableau("Classic Orange-White-Blue Light", limits = c(-1,1)) +
  theme(axis.title = element_text()) +
  ylab("Feature") +
  xlab("Feature") +
  labs(title = "Correlation Matrix")
```

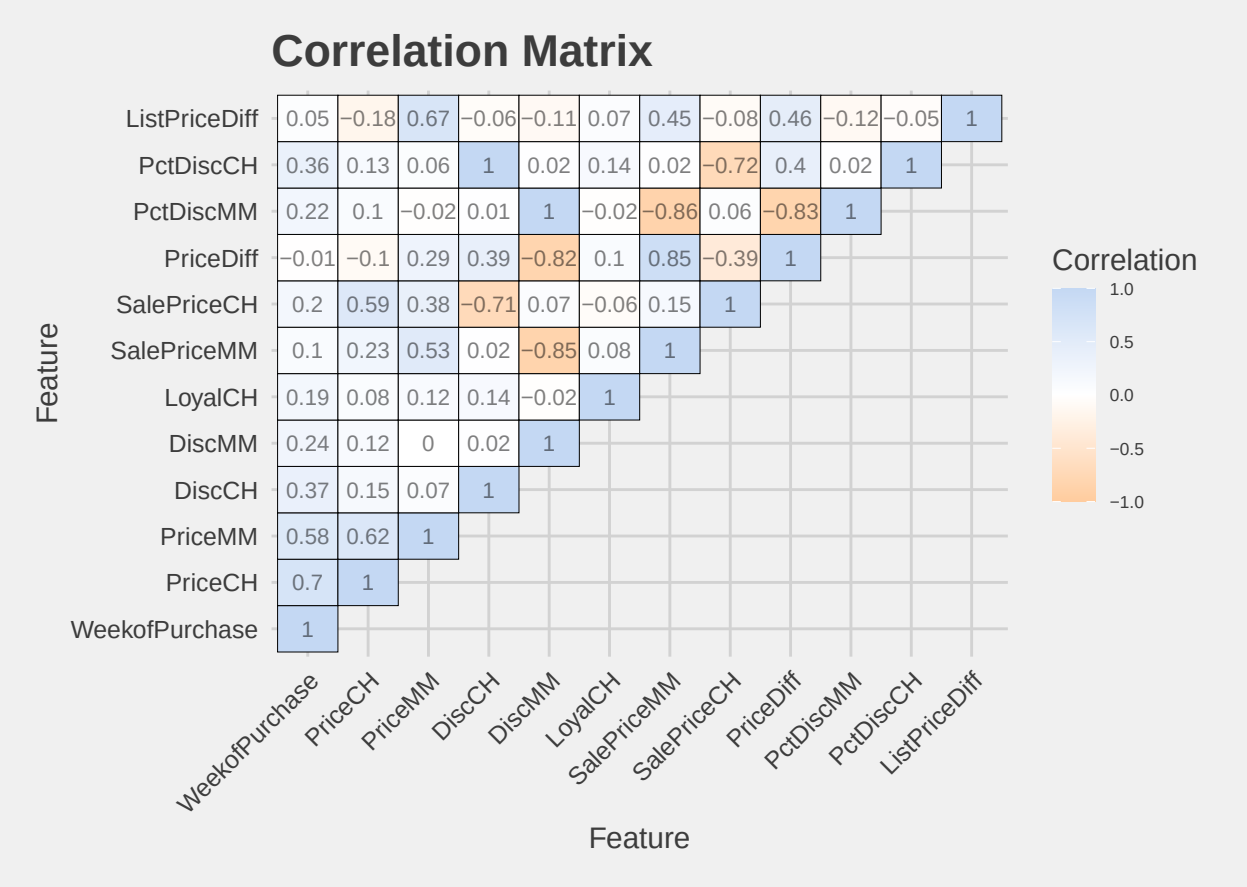

Figure 1: Correlation Matrix pre-engineering

```
rm(Integers, melted_cormat, nums)
```

Removal and Recoding of Variables

As shown in figure 1, there are a number of relationships that show a correlation above our threshold of absolute 0.7. First, there are a number of variables that can either be inferred from another set of variables or that give the exact same information: pricediff variables and the disc variables. Pricediff can be inferred from SalePriceMM and SalePriceCH (formula= SalePriceMM - SalePriceCH) and therefore is strongly correlated with both variables. Both SalePrice variables add more information to the data set than Pricediff and therefore PriceDiff will be removed. The same reasoning applies to ListPriceDiff and therefore will also be removed. For the Disc (discount) variables, the choice is the same. The height of the discount can be inferred by subtracting the ListPrice from the Price and therefore both Disc and PctDisc values can be calculated. The information held by the variables can be found mostly in the fact if a discount is present and not the height of the discount. Knowing this, two decisions are made. First, create a new variable called "Discount" that shows if a discount is present or not and, if so, for which brand a discount is present. Second, remove the four old discount variables (pctDisc and Disc) While it can be debated whether WeekofPurchase should be a factorized variable or a numeric variable, the fact that it holds a progressive set of numbers in which the progression holds information (the flow of time), we opt to encode WeekofPurchase as numeric. Encoding WeekofPurchase as numeric shows a strong correlation with PriceCH of exactly 0.7, meaning the variable should be removed.

```
df$Discount <- factor(ifelse(df$DiscMM>0,"MM",
                             ifelse(df$DiscCH>0,"CH","None")),
                      levels=c("None", "CH","MM"))

df <- df[,-c(2,6,7,13,15,16,17)] #Removing aforementioned Variables
```

Correlation Matrix Post-Engineering

```
nums <- unlist(lapply(df, is.numeric))
Integers <- df[ , nums]

melted_cormat <- melt(cor(Integers))

names(melted_cormat)[3] <- "Correlation"

melted_cormat <- melted_cormat %>%
    arrange(Var1) %>%
    group_by(Var1) %>%
    filter(row_number() >= which(Var2 == Var1))

ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=Correlation)) +
  geom_tile(aes(fill=Correlation), col = "Black") +
  theme_fivethirtyeight () +
  theme(legend.position="right",legend.direction = "vertical",
        legend.text = element_text(size=7)) +
  geom_text(label = round(melted_cormat$Correlation, digits = 2),
            color="Black", size=3, alpha=0.5) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust=1, size=10),
        axis.text.y = element_text(angle=0,vjust = 0.5, hjust=1, size=10)) +
```

```
  scale_fill_gradient2_tableau("Classic Orange-White-Blue Light", limits = c(-1,1)) +
  theme(axis.title = element_text()) +
  ylab("Feature") +
  xlab("Feature") +
  labs(title = "Correlation Matrix")
```

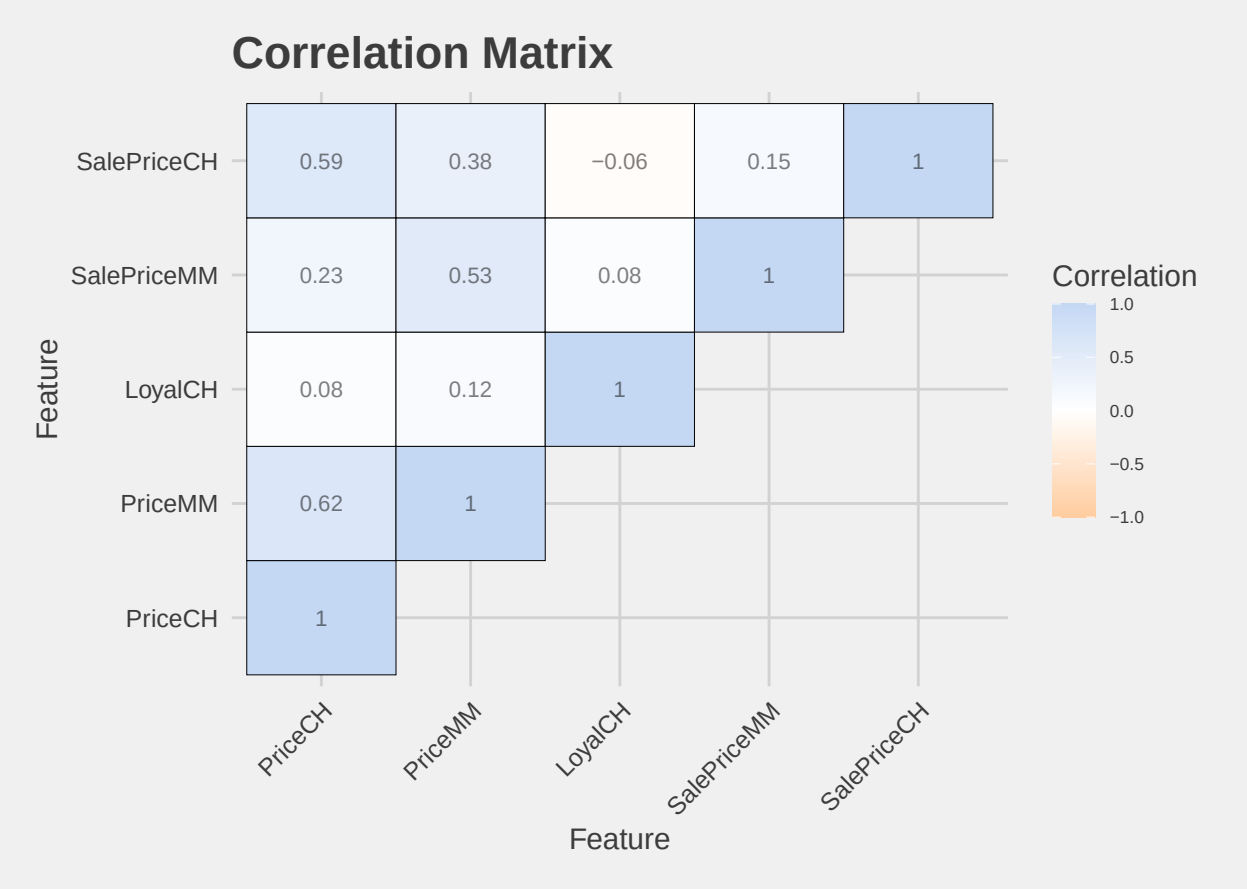

Figure 2: Correlation Matrix post-engineering

```
rm(Integers, melted_cormat, nums)
```

As can be seen in figure 2, no correlations above 0.7 exist anymore which is expected to eliminate multicollinearity. To check this, VIF scores will be created for the appropriate models in the machine learning section.

Factorized Variables

Now that we engineered the numeric variables, we take a look at the factorized variables to combat multicollinearity and singularity (correlation of 1). First, there are the store variables which seem to hold identical information and therefore only one of them should be retained. More specifically, we are talking about StoreID, Store7 and STORE. Since StoreID is the most complete (includes store7) and optimally leveled (Store 1,2,3,4,7), we will retain StoreID and remove Store7 and STORE.

For “Special”, we will do the same as we did for Discount. While this has no effect due to future dummy encoding, it does make the data set more concise without losing information, which in turn makes plotting relationships easier.

```
df$Special <- factor(ifelse(df$SpecialMM==1,"MM",
                            ifelse(df$SpecialCH==1,"CH","None")),
                     levels=c("None", "CH","MM"))

df <- df[,-c(5,6,10,11)]
```

Outlier Detection and Relationships: Numerical Variables

```
num_list <- colnames(df[sapply(df, is.numeric)])
plot_list <- list(1)

counter = 0

for (i in num_list) {

  counter = counter+1
  x = ggplot(data=df, aes_string(x = "Purchase", y = i)) +
  geom_boxplot() + ggtitle(paste0("Purchase vs ", as.character(i), sep= " ")) +
    theme_fivethirtyeight() + theme(plot.title = element_text(size=10))


  assign(paste0("plt",i), x)
  plot_list <- rlist::list.append(plot_list, x)
}

plot_list <- plot_list[-1]

n <- length(plot_list)
nCol <- floor(sqrt(n))

grid.arrange(grobs=plot_list, widths = c(1,1), ncol=nCol, layout_matrix =  rbind(c(1,1),
                                                                                 c(2,3),
                                                                                 c(4,5)),
             top = "Relationships between Target and Numeric Variables")
```

```
rm(plot_list, counter, num_list, x, i, n, nCol)
```

As stated in the Preliminary Analysis section of this paper, the ranges of the continuous variables are not abnormal therefore it seems there are not any data entry errors present in the data set that need cleaning. After the initial assessment of the range, we use boxplots to determine where the outliers are present in the data set as shown in figure 3. Boxplots show an easy overview and allow for interpretation of categorized outliers based on the 1.5 interquartile range threshold. Regarding outliers, it is important to interpret the data before removing anything as an outlier can properly represent the population in a sample. Removing the outlier would then misrepresent the population making results less generalizable.

PriceMM shows two outlier values at a Price of 1.69 and 1.79. These two values do not seem abnormal, as some customers simply prefer CH over MM and price has little effect on their purchasing behavior, therefore this is no cause for removal of these data points.

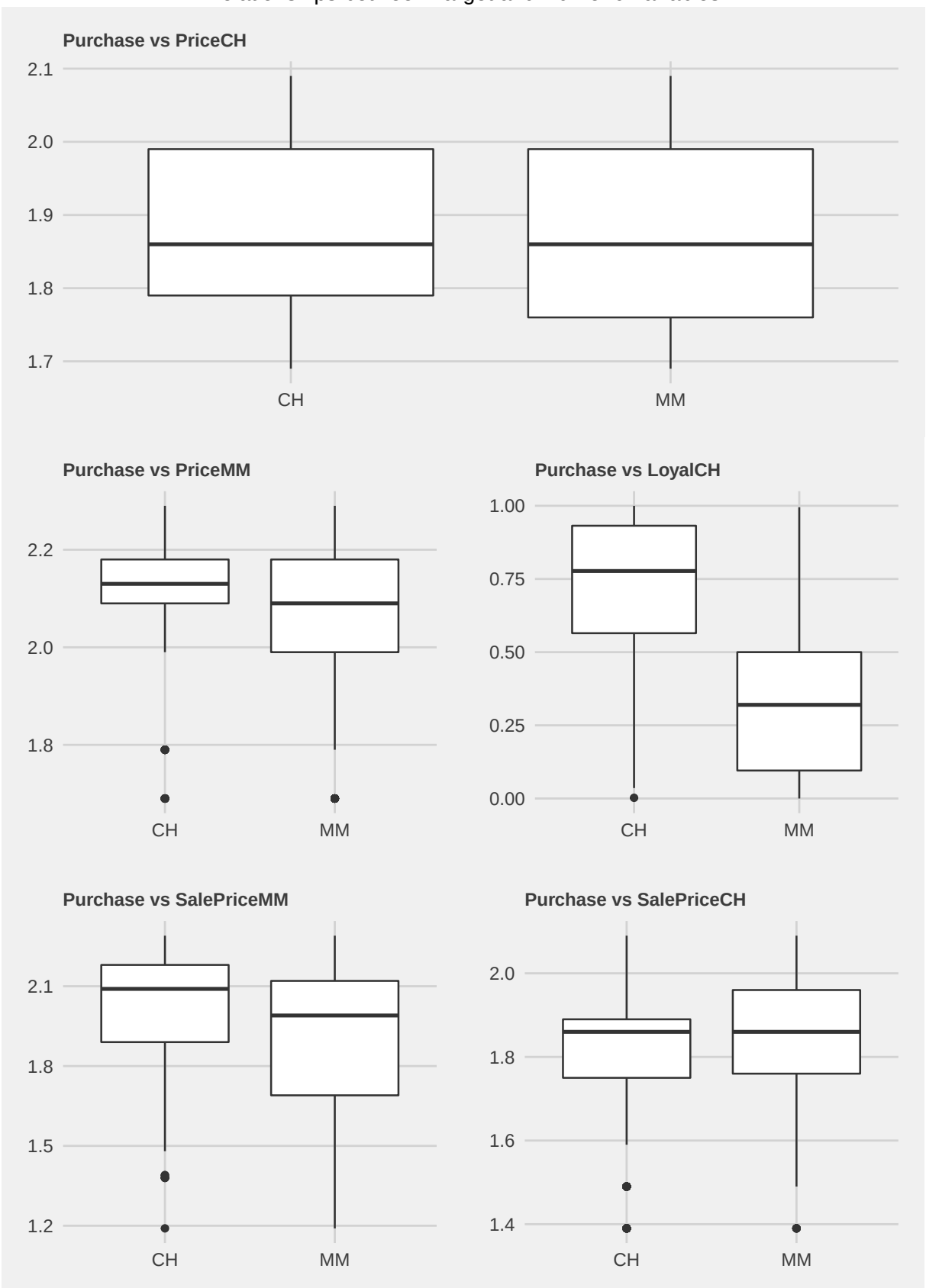

Figure 3: Relationships between Target and Numeric Variables

LoyalCH shows one outlier at 0.002443. A loyalty of 0.002443 but buying CH over MM is odd, but explainable. Imagine the customer being in a hurry and not having time to search for MM. It should also be noted that the interquartile range is very wide, making it a more common occurrence buying CH with low CH loyalty. With the reasons given, the data entry should not be removed.

SalePriceMM No outliers will be removed for the same reason as PriceMM.

SalePriceCH No outliers will be removed for the same reason as Price MM, some customers simply prefer a brand and are not affected by price within the range of the dataset.

Apart from Identifying outliers, this plot also shows relationships between the target variable and the numeric independent variables within the data set:

1. The Price of MM seems to have a larger influence on customer behavior than the Price of CH. As can be seen in the above figure, the difference in the mean of CH and MM purchases based on the Price of MM is quite a lot larger then when looking at the price of CH, with a higher price of MM causing more CH sales.
2. Higher values of LoyalCH are cause for more CH sales over MM sales

Relationships: Categorical Variables

```
fac_list <- colnames(df[sapply(df, is.factor)])
fac_list <- fac_list[-1]
plot_list <- list(1)

counter = 0

for (i in fac_list) {
  counter = counter+1

  x = ggplot(data=df, aes_string(x = i,fill = "Purchase")) +
  geom_bar(position = "dodge") +
  ggtitle(paste0("Purchase vs ", as.character(i), sep= " ")) +
  theme_fivethirtyeight() +
  theme(plot.title = element_text(size=10), legend.position = ("none")) +
  scale_fill_tableau() +
  geom_text(stat='count', aes(label=Purchase),
            vjust=-1.5, position = position_dodge(width = 1), size=3) +
    geom_text(stat='count', aes(label=..count..), color="grey25",
            vjust=1, position = position_dodge(width = 1), size=3) +
  scale_y_continuous(limits = c(0, 1.01 * sum(ifelse(df$Purchase=="CH", 1, 0))))

  assign(paste0("plt",i), x)

  plot_list <- rlist::list.append(plot_list, x)
}

plot_list <- plot_list[-1]

n <- length(plot_list)
nCol <- floor(sqrt(n))
```

```
grid.arrange(grobs=plot_list, ncol=nCol, layout_matrix =  rbind(c(1),c(2),c(3)),
             top = "Relationships between Target and Categorical Variables")
```

```
rm(plot_list, counter, fac_list, x, i, n, nCol)
```

Figure 4 is included to find abnormalities within the relationship between independent categorical variables and the target variable. This plot will help understand the distribution of the data and for better interpretation of the logistic regression results. Some observations;

1. store 7 is selling a large amount more CH then any other store.
2. As expected, customers default to the CH brand when no discount is available due to its generally lower price, creating the expectation that price should have a significant influence on the target variable. The same relationship between special and CH shows, giving the same expectation between CH and price.

Normalization and Scaling

In this project, no scaling of variables has been performed. This is due to the magnitude of the variables not varying to the degree that one variable will dominate others, making scaling not necessary. Normalization has not been performed as multiple different transformations did not seem to improve the linear relationship between the predictor variables and the logit of Logistic regression. Next to this, SVM has no linearity assumption at all.

Dummy Encoding

Dummy encoding is performed to prepare the variables for machine learning. Dummy encoding is used to remove the interpretation the higher categorical levels are better when this is not the case. Take StoreID for example, store 1 is not worse than Store 7, however if we do not Dummy encode, the algorithm will interpret it that way. By removing this interpretation, the algorithms will be able to perform better, hence dummy encoding is used in this project.

```
df_target_bu <- df$Purchase
df_bu <- df

df <- dummyVars(Purchase~., data=df, fullRank = TRUE) %>%
  predict(newdata = df)
```

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

```
df <- as.data.frame(df)
df$Purchase <- df_target_bu


rm(df_target_bu, df_bu)
```

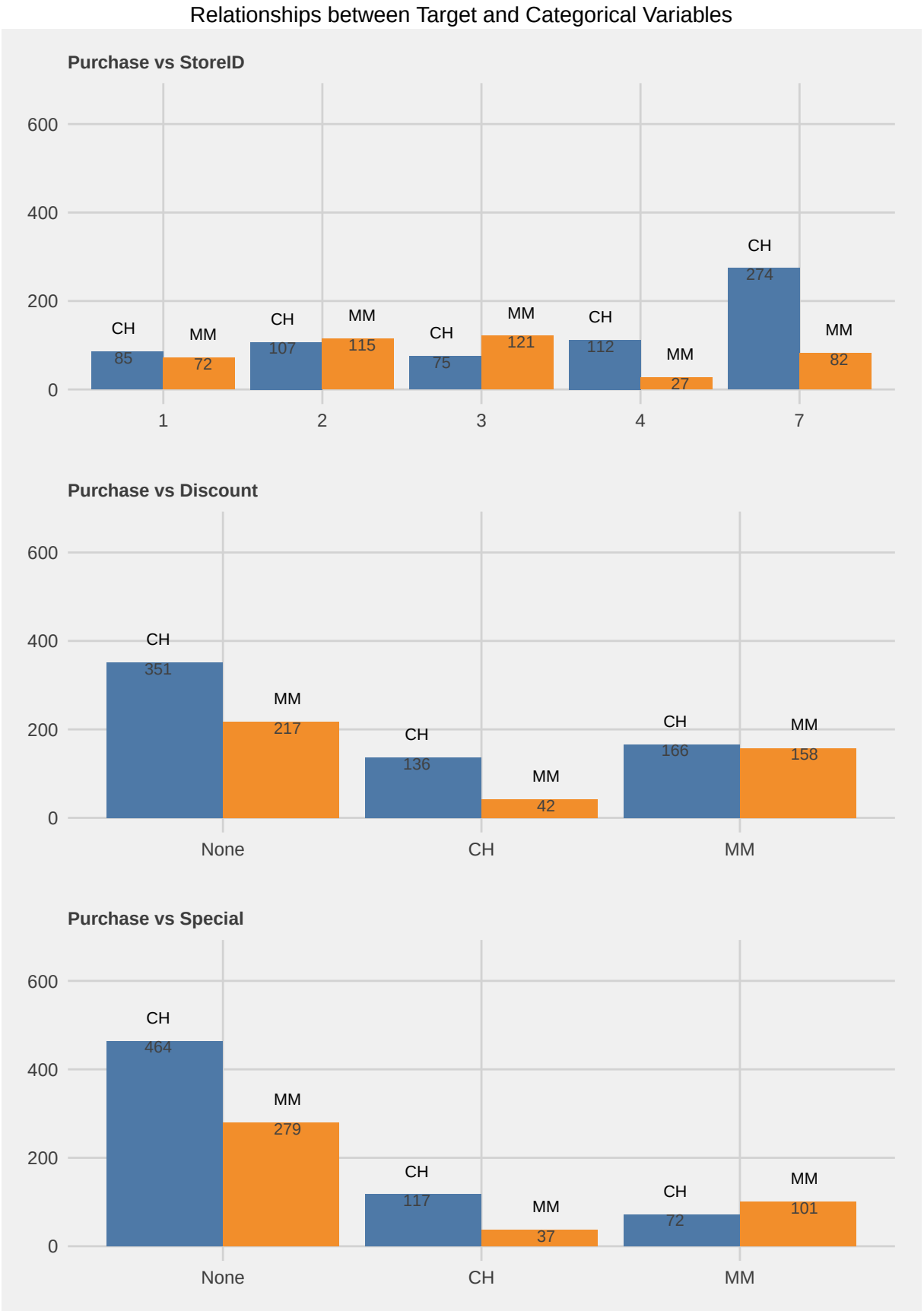

Figure 4: Relationships between Target and Categorical Variables

Hold-Out Evaluation

We use a hold-out evaluation to be able to generate out-of-sample performance metrics. Out-of-sample performance metrics are needed to test for overfitting of models and to find the predictive power of a model on new data. Taking into account the goal of the sales manager, it is important to be able to properly assess out-of-sample predictive power as an overfit model has no use in practice.

```
set.seed(500)
inTrain <- createDataPartition(y=df$Purchase, p = 0.75, list=FALSE)
train <-df[inTrain,-length(names(df))]
train_target<-df[inTrain,length(names(df))]
test <-df[-inTrain,-length(names(df))]
test_target<-df[-inTrain,length(names(df))]
```

Machine Learning

Set-up Metric DataFrame

```
resultsclassification <- data.frame(Model = as.character(),
           Sample = as.character(),
           ACC = numeric(),
           TPR = numeric(),
           Precision = numeric(),
           F1 = numeric())


resultsclassification <- rbind(resultsclassification, c("Test", "Test", 1, 1, 1, 1, 1, 1))
names(resultsclassification) <- c("Model","Sample","ACC", "TPR_CH",
                                  "TPR_MM", "PRECISION_CH", "PRECISION_MM", "F1")

resultsclassification$Model <- as.character(resultsclassification$Model)
resultsclassification$Sample <- as.character(resultsclassification$Sample)
resultsclassification[sapply(resultsclassification, is.factor)] <-
  lapply(resultsclassification[sapply(resultsclassification, is.factor)], as.numeric)
```

Logistic Regression

```
system.time(glm_model <- glm(train_target~., data=train, family = binomial))
```

```
##    user  system elapsed
##       0       0       0
```

```
summary(glm_model)
```

```
##
## Call:
## glm(formula = train_target ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8320  -0.5371  -0.2247   0.5371   2.8680
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.3565     2.2319   1.504  0.13262
## StoreID.2    -0.1348     0.3196  -0.422  0.67312
## StoreID.3    -0.1623     0.3839  -0.423  0.67236
## StoreID.4    -0.2748     0.4416  -0.622  0.53370
## StoreID.7    -0.8406     0.3334  -2.521  0.01170 *
## PriceCH      -1.3603     2.0721  -0.656  0.51151
## PriceMM       0.2843     1.4376   0.198  0.84325
## LoyalCH      -6.3446     0.4750 -13.358  < 2e-16 ***
## SalePriceMM  -3.3958     1.0765  -3.155  0.00161 **
## SalePriceCH   4.6369     1.4684   3.158  0.00159 **
```

```
## Discount.CH   0.3323     0.4293   0.774  0.43883
## Discount.MM  -0.5804     0.4684  -1.239  0.21533
## Special.CH    0.3743     0.3853   0.972  0.33121
## Special.MM    0.3567     0.3306   1.079  0.28071
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1073.9  on 802  degrees of freedom
## Residual deviance:  610.0  on 789  degrees of freedom
## AIC: 638
##
## Number of Fisher Scoring iterations: 5
```

```
metric_list <- c("ACC", "TPR", "PRECISION", "F1")

#Train
prediction_glm_train <- predict(glm_model, train)
metrics_train <- round(mmetric(train_target, prediction_glm_train,metric_list),2)

#Test
prediction_glm_test <- predict(glm_model, test)
metrics_test <- round(mmetric(test_target, prediction_glm_test,metric_list),2)

trainset <- prepend(unname(metrics_train), c("GLM","In-sample"))
```

```
## Warning: `prepend()` is deprecated as of rlang 0.4.0.
##
## Vector tools are now out of scope for rlang to make it a more
## focused package.
## This warning is displayed once per session.
```

```
testset  <- prepend(unname(metrics_test), c("GLM", "Out-of-sample"))

resultsclassification <- rbind(resultsclassification, trainset)
resultsclassification <- rbind(resultsclassification, testset)

resultsclassification <- resultsclassification[-1,]
```

Validate Assumptions

Multicolinearity Below we can see the VIF scores of the logistic regression model's independent variables. The lower the VIF, the better and SalePriceMM has a VIF score that is quite high. However, as long as VIF is lower than 10, there is no cause for major concern (Glen, 2020). Given the VIF scores, the Logistic Regression model passes the multicolinearity assumption.

```
car::vif(glm_model)
```

```
##   StoreID.2   StoreID.3   StoreID.4   StoreID.7     PriceCH     PriceMM
##    1.835915    2.166924    1.843809    2.091178    4.663706    4.139506
##     LoyalCH SalePriceMM SalePriceCH Discount.CH Discount.MM  Special.CH
```

```
##    1.159548    7.434566    3.879728    2.012180    4.838164    1.768947
##  Special.MM
##    1.639445
```

```
numeric_list <- c(names(train)[c(5,6,7,8,9)])

idx <- match(numeric_list, names(train))

Linplot <- train[,idx]

prob <- predict(glm_model, train, type="response")

Linplot <- Linplot %>%
  mutate(logit = log(prob)/(1-prob)) %>%
  gather(key = "predictors", value =  "predictor.value", -logit)

ggplot(Linplot, aes(logit, predictor.value)) +
  geom_point(size = 0.5, alpha = 0.5) +
  geom_smooth() +
  theme_fivethirtyeight() +
  facet_wrap(~predictors, scales = "free_y") +
  ggtitle("Relationship between Logit and Continious independent variables") +
  theme(plot.title = element_text(size = 5))
```

Linearity

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

The above code is taken from (Kassambara, 2018) and used to create figure 5. As stated before, while some variables do not show a great deal of linearity, trying to transform these variables did not improve the linearity between the variables and the logit. While the assumption is not perfectly adhered to, it seems not to prevent us from fulfilling the goals of this project

Step-Wise Logistic Regression

Since Logistic regression's performance is heavily dependent on the data that the model is fed, it is interesting to see how a step-wise regression would perform compared to the base model and if the MASS package will remove any variables.

```
step_model <- glm_model %>% stepAIC(trace = FALSE)

summary(step_model)
```

```
##
## Call:
## glm(formula = train_target ~ StoreID.7 + LoyalCH + SalePriceMM +
##     SalePriceCH, family = binomial, data = train)
##
## Deviance Residuals:
```

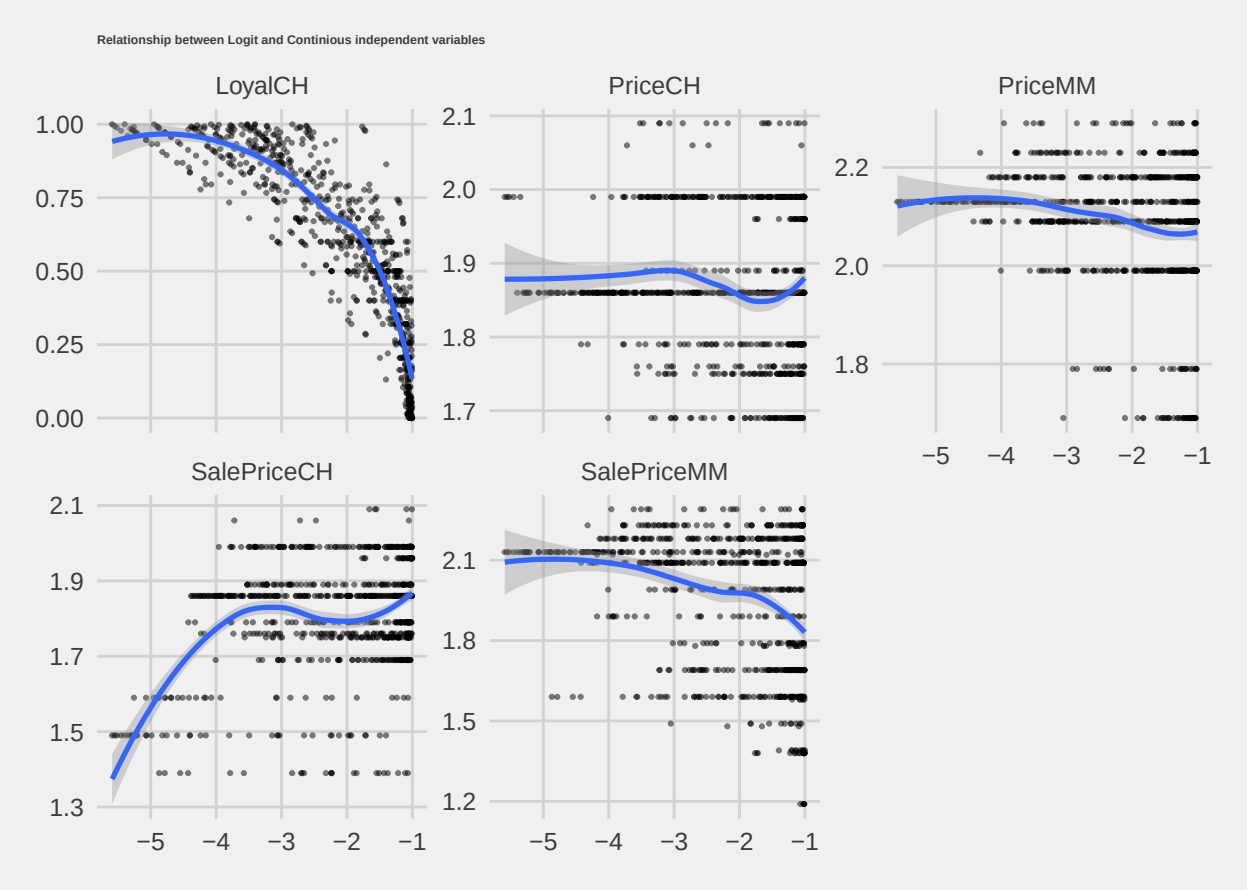

Figure 5: Relationship between Logit and Continious independent variables

```
##     Min       1Q   Median       3Q      Max
## -2.8277  -0.5334  -0.2296   0.5422   2.6600
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.0523     1.5099   2.021 0.043229 *
## StoreID.7    -0.7889     0.2423  -3.256 0.001131 **
## LoyalCH      -6.3435     0.4548 -13.949  < 2e-16 ***
## SalePriceMM  -2.7650     0.4112  -6.725 1.76e-11 ***
## SalePriceCH   2.9853     0.7862   3.797 0.000146 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1073.9  on 802  degrees of freedom
## Residual deviance:  614.8  on 798  degrees of freedom
## AIC: 624.8
##
## Number of Fisher Scoring iterations: 5
```

```
#Train
prediction_step_train <- predict(step_model, train)
metrics_train <- round(mmetric(train_target, prediction_step_train,metric_list),2)

#Test
prediction_step_test <- predict(step_model, test)
metrics_test <- round(mmetric(test_target, prediction_step_test,metric_list),2)

trainset <- prepend(unname(metrics_train), c("StepWiseGLM","In-sample"))
testset  <- prepend(unname(metrics_test), c("StepWiseGLM", "Out-of-sample"))

resultsclassification <- rbind(resultsclassification, trainset)
resultsclassification <- rbind(resultsclassification, testset)
```

Validate Assumptions

Multicolinearity Below we can see the VIF scores of the Step-wise logistic regression model's independent variables. As we can see, the VIF scores went down considerably all, however around 1 right now which is a near perfect score. A VIF of 1 means no correlation is detected between predictors at all and our scores are close to optimal in that sense.

```
car::vif(step_model)
```

```
##   StoreID.7     LoyalCH SalePriceMM SalePriceCH
##    1.104447    1.077496    1.120432    1.164500
```

Linearity As expected by the small increase in accuracy, the linearity plots are (close to) the same as the logistic regression model and will therefore not be shown again. The same reasoning and conclusion applies to the step-wise model as it did to the logistic regression model.

SVM

Since support vector machine is dependent on hyperparameter inputs we utilize gridsearching with a 5-fold cross validation to find the optimal hyperparameters.

```
Tcontrol <- tune.control(cross = 5, sampling = "cross")

svm_tune <- tune(svm, train.x = train, train.y = train_target, kernel = "radial",
                 tunecontrol = Tcontrol, scale = FALSE,
                 ranges = list(epsilon = seq(0,10,2), cost=seq(0.0001,10,2)))

paste0("Best Epsilon = ",svm_tune$best.parameters$epsilon,
       " | Best Cost ", round(svm_tune$best.parameters$cost,2))
```

```
## [1] "Best Epsilon = 0 | Best Cost 2"
```

```
svm_tune <- tune(svm, train.x = train, train.y = train_target, kernel = "radial",
                 tunecontrol = Tcontrol, scale = FALSE,
                 ranges = list(epsilon = seq(ifelse(svm_tune$best.parameters$epsilon-2<=0,0,
                                                    svm_tune$best.parameters$epsilon-2),
                                             svm_tune$best.parameters$epsilon+2,1),
                               cost=seq(svm_tune$best.parameters$cost-2,
                                        svm_tune$best.parameters$cost+2,1)))

paste0("Best Epsilon = ",svm_tune$best.parameters$epsilon,
       " | Best Cost ", round(svm_tune$best.parameters$cost,2))
```

```
## [1] "Best Epsilon = 0 | Best Cost 4"
```

```
svm_tune <- tune(svm, train.x = train, train.y = train_target, kernel = "radial",
                 tunecontrol = Tcontrol, scale = FALSE,
                 ranges = list(epsilon = seq(ifelse(svm_tune$best.parameters$epsilon-1<=0,0,
                                                    svm_tune$best.parameters$epsilon-1),
                                             svm_tune$best.parameters$epsilon+1,.5),
                               cost=seq(svm_tune$best.parameters$cost-1,
                                        svm_tune$best.parameters$cost+1,.5)))

paste0("Best Epsilon = ",svm_tune$best.parameters$epsilon,
       " | Best Cost ", round(svm_tune$best.parameters$cost,2))
```

```
## [1] "Best Epsilon = 0 | Best Cost 5"
```

```
svm_tune <- tune(svm, train.x = train, train.y = train_target, kernel = "radial",
                 tunecontrol = Tcontrol, scale = FALSE,
                 ranges = list(epsilon = seq(ifelse(svm_tune$best.parameters$epsilon-.5<=0,0,
                                                    svm_tune$best.parameters$epsilon-.5),
                                             svm_tune$best.parameters$epsilon+.5,.1),
                               cost=seq(svm_tune$best.parameters$cost-.5,
                                        svm_tune$best.parameters$cost+.5,.1)))

paste0("Best Epsilon = ",svm_tune$best.parameters$epsilon,
       " | Best Cost ", round(svm_tune$best.parameters$cost,2))
```

```
## [1] "Best Epsilon = 0 | Best Cost 5.4"
```

```
(svm_model <- svm_tune$best.model)
```

```
##
## Call:
## best.tune(method = svm, train.x = train, train.y = train_target,
##     ranges = list(epsilon = seq(ifelse(svm_tune$best.parameters$epsilon -
##         0.5 <= 0, 0, svm_tune$best.parameters$epsilon - 0.5), svm_tune$best.parameters$epsilon +
##         0.5, 0.1), cost = seq(svm_tune$best.parameters$cost - 0.5,
##         svm_tune$best.parameters$cost + 0.5, 0.1)), tunecontrol = Tcontrol,
##     kernel = "radial", scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  5.4001
##
## Number of Support Vectors:  362
```

```
metric_list <- c("ACC", "TPR", "PRECISION", "F1")
#Train
prediction_svm_train <- predict(svm_model, train)
metrics_train <- round(mmetric(train_target, prediction_svm_train,metric_list),2)

#Test
prediction_svm_test <- predict(svm_model, test)
metrics_test <- round(mmetric(test_target, prediction_svm_test,metric_list),2)

trainset <- prepend(unname(metrics_train), c("SVM","In-sample"))
testset  <- prepend(unname(metrics_test), c("SVM", "Out-of-sample"))

resultsclassification <- rbind(resultsclassification, trainset)
resultsclassification <- rbind(resultsclassification, testset)
```

Since SVR makes no assumptions other than independence of data, nothing will be discussed here. The final performance scores can be found below in section Results and Conclusion

Results and Conclusion

Goal 1: Variable Selection

```
summary(step_model)
```

```
##
## Call:
## glm(formula = train_target ~ StoreID.7 + LoyalCH + SalePriceMM +
##     SalePriceCH, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8277  -0.5334  -0.2296   0.5422   2.6600
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.0523     1.5099   2.021 0.043229 *
## StoreID.7    -0.7889     0.2423  -3.256 0.001131 **
## LoyalCH      -6.3435     0.4548 -13.949  < 2e-16 ***
## SalePriceMM  -2.7650     0.4112  -6.725 1.76e-11 ***
## SalePriceCH   2.9853     0.7862   3.797 0.000146 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1073.9  on 802  degrees of freedom
## Residual deviance:  614.8  on 798  degrees of freedom
## AIC: 624.8
##
## Number of Fisher Scoring iterations: 5
```

```
x <- resultsclassification[c(3,4),]
row.names(x) <- NULL


print(xtable(x,
             caption="GLM Model Performance across different metrics"), comment = FALSE)
```

	Model	Sample	ACC	TPR_CH	TPR_MM	PRECISION_CH	PRECISION_MM	F1
1	StepWiseGLM	In-sample	81.69	91.22	66.77	81.13	82.94	85.88
2	StepWiseGLM	Out-of-sample	82.77	92.64	67.31	81.62	85.37	86.78

Table 1: GLM Model Performance across different metrics

After Running the Logistic step-wise regression, which eliminates variables based on best model-fit, we can see in the output above that 4 variables remain: StoreID.7, LoyalCH, SalePriceMM and SalePriceCH. These four variables have a AIC (Akaike information criterion) of 624.8 over 638 when running the full model with logistic regression indicating a better fit (lower is better), therefore the step-wise logistic regression model is the model we will draw our conclusions from.

While a lot of information is shown in the above output, the most important section are the Coefficients, their P-value, and their Estimates. The coefficient are variables names included in the model, the estimates are the magnitude and direction of the effect they have on customer behavior, and the P-value shows if the finding is statistically significant. As can be seen above, all coefficients are statistically significant to a level of $p<0.01$. This is due to the elimination process of step-wise logistic regression. When looking at the Logistic regression output in appendix C, it shows more variables which do not have a statistically significant contribution to the model and should therefore not be considered when determining strategies to influence customer behavior. While step-wise logistic regression does not eliminate all non-significant contributions to the model per se, it does however reduce clutter and show a more focused view of the most important predictors in the process often eliminating most non significant contributions.

Due to the low P-Values and the accuracy of over 82.77%, as shown in table 1 we can be confident in our model's ability to show generalizable relationships regarding customer behavior. The accuracy of 82.77% percent is satisfactory given the amount of data giving an accuracy increase of around 20% over the majority predictor benchmark. Data collection should continue however, in the future, the analysis should be rerun with a bigger sample to create an even more robust result. Being right 80% of the time gives enough foundation to start implementing tactics based on the model output which can be adapted based on future analyses when more information is available including performance metrics of the newly implemented tactics.

So with all of our contributions being significant, we can start interpreting the coefficients and estimates in the output. The estimate the output shows are logits or log-odds (log(odds)). Interpreting log-odds is hard due to it being non-intuitive, therefore we use the function below (Sauer, 2017) to transform to probabilities.

```
logit2prob <- function(logit){
  odds <- exp(logit)
  prob <- odds / (1 + odds)
  return(prob)
}

paste0("Intercept gives a probability of buying MM of : ",
       round(logit2prob(coef(step_model)[1]),4)*100, "%")
```

```
## [1] "Intercept gives a probability of buying MM of : 95.49%"
```

```
paste0("Intercept and yes to StoreID.7 gives a probability of buying MM of: ",
       round(logit2prob(coef(step_model)[1] + 1 * coef(step_model)[2]),4)*100, "%")
```

```
## [1] "Intercept and yes to StoreID.7 gives a probability of buying MM of: 90.58%"
```

```
paste0("Intercept and one unit increase of LoyalCH gives a probability of buying MM of: ",
       round(logit2prob(coef(step_model)[1] + 1 * coef(step_model)[3]),4)*100, "%")
```

```
## [1] "Intercept and one unit increase of LoyalCH gives a probability of buying MM of: 3.59%"
```

```
paste0("Intercept and one unit increase of SalePriceMM gives a probability of buying MM of: ",
       round(logit2prob(coef(step_model)[1] + 1 * coef(step_model)[4]),4)*100, "%")
```

```
## [1] "Intercept and one unit increase of SalePriceMM gives a probability of buying MM of: 57.13%"
```

```
paste0("Intercept and one unit increase of SalePriceCH gives a probability of buying MM of: ",
       round(logit2prob(coef(step_model)[1] + 1 * coef(step_model)[5]),4)*100, "%")
```

```
## [1] "Intercept and one unit increase of SalePriceCH gives a probability of buying MM of: 99.76%"
```

```
paste0("Intercept and one unit increase of SalePriceCH and PriceMM gives a MM prob of: ",
       round(logit2prob(coef(step_model)[1] + 1 *
                          coef(step_model)[5] + 1*coef(step_model)[4]),4)*100, "%")
```

```
## [1] "Intercept and one unit increase of SalePriceCH and PriceMM gives a MM prob of: 96.35%"
```

As can be seen above in the output, the intercept of the model gives a base probability of buying Minute Maid of 95.49%. This base probability will either increase or decrease based on what the predictor values will be. Negative estimates will lower the probability of Minute Maid per unit increase of the predictor variable and positive estimates will increase the the probability of Minute Maid per unit increase of the predictor variable. Above is shown the effect on the base probability of buying Minute Maid of a one unit increase per predictor variable. Going over each variable we will look at the effect and apply it to a real life scenario

1. **StoreID.7**: As shown StoreID.7 has a negative estimate and will therefore lower the probability of buying Minute Maid when it is 1. In other words, customers are less likely to buy Minute Maid in store 7 compared to store 1,2,3,4. It should be investigated why. One possible reason can be the store layout or the location of Minute Maid on the shelves compared to Citrus Hill. If Minute Maid is on eye level in store 1,2,3,4 and not in store 7 for example it can have an effect on people buying more Citrus Hill simply because they do not see Minute Maid. It is important to find out the differences in how Minute Maid is presented between store 1,2,3,4 and store 7 to be able to increase relative Minute Maid sales compared to Citrus Hill in store 7.
2. **LoyalCH**: LoyalCH has a large native estimate and is therefore a strong influence in people buying Citrus Hill over Minute Maid. This makes sense due to this variable being constructed based on prior customer behavior and it showing the loyalty a customer has towards Citrus Hill as a brand. When LoyalCH is one, the highest value it can assume, the probability of buying Minute Maid is 3.59% over a base of 95.49%. The drastic decrease between probabilities shows the influence this variable has on the choice of buying CH over MM and shows the importance into lowering customer brand loyalty towards CH. Lowering brand loyalty of CH and increasing MM brand loyalty can be achieved by using marketing tactics for Minute Maid. For example allowing customers to taste the difference between MM and CH to show why Minute Maid would be better or tastier. Advertising works as well, where people are reminded of competitors of CH and are urged to expand their horizon regarding Orange Juice. Offering discounts on Minute Maid to urge people to try Minute Maid and to possibly make them switch is also a viable strategy.
3. **SalePrice**: Both SalePriceCH and SalePriceMM are interconnected variables since they have reversing effects per unit of increase where SalePriceCH has a positive effect on buying Minute maid and SalePriceMM has a negative effect on buying MinuteMaid. This makes sense, where if either brand becomes more expensive customers tend to switch to the other brand. When looking at magnitude however, it can be seen that SalePriceCH has a slightly larger absolute estimate than SalePriceMM does, meaning that customers are willing to switch to MM from CH faster based on price then they would switch to CH from MM. It should be noted that the SalePrice encompassed more than just the price as it is the final price. The SalePrice also holds information regarding Discounts, as in the final price matters over the list price. Discounts can influence the SalePrice and therefore have an effect on customer behavior of buying Minute Maid over Citrus Hill.

To summarize, the brand manager should make sure Minute Maid is displayed the same in Store 7 as in Store 1,2,3 and 4. The brand manager should also try to show customers the added value of buying Minute Maid over Citrus Hill so customers can justify the higher price they will pay for Minute Maid. To help people switch, discounts should be offered so people are enticed to try Minute Maid. This will hopefully make them stick to Minute Maid due to superior quality, lowering both the SalePriceMM and LoyalCH and boosting overall Minute Maid sales.

Goal 2: Forecasting Performance

```
row.names(resultsclassification) <- c(1:nrow(resultsclassification))

resultsclassification %>% ggplot(aes(x=reorder(Model, as.numeric(ACC)),y=as.numeric(ACC),
                                     group=Sample, fill=Sample)) +
  geom_bar (stat="identity", position = position_dodge(width = .75),
            color="grey25", size=.1) +
  theme_fivethirtyeight() + scale_fill_tableau() +
  labs(title = "Classification Model Performance", subtitle = "Higher is better") +
  theme(axis.title = element_text()) +
  ylab("Accuracy Score") + xlab("Classification Models") +
  scale_y_continuous(labels = function(x) paste0(x, "%")) +
  geom_label(aes(label=paste0(ACC, " %")),fill="white", vjust=-1,
             position = position_dodge(width = .85)) +
  coord_cartesian(ylim=c((floor(min(as.numeric(resultsclassification$ACC))/ 5)-1)*5,
                         (ceiling(max(as.numeric(resultsclassification$ACC))/ 5)+1)*5))
```

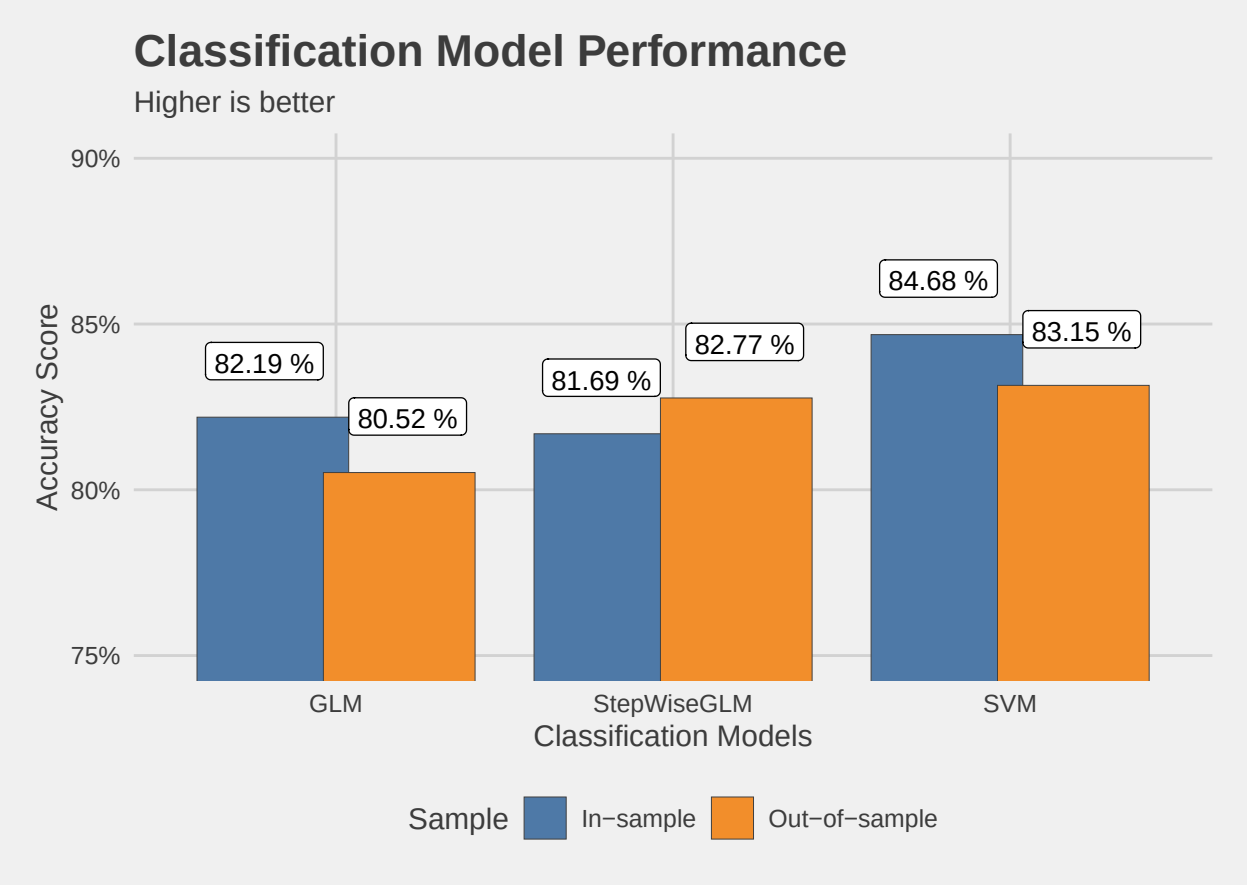

Figure 6: Classification Model Performance

```
print(xtable(resultsclassification,
             caption="Model Performance across different metrics"), comment = FALSE)
```

	Model	Sample	ACC	TPR_CH	TPR_MM	PRECISION_CH	PRECISION_MM	F1
1	GLM	In-sample	82.19	91.63	67.41	81.49	83.73	86.26
2	GLM	Out-of-sample	80.52	91.41	63.46	79.68	82.5	85.14
3	StepWiseGLM	In-sample	81.69	91.22	66.77	81.13	82.94	85.88
4	StepWiseGLM	Out-of-sample	82.77	92.64	67.31	81.62	85.37	86.78
5	SVM	In-sample	84.68	88.78	78.27	86.48	81.67	87.61
6	SVM	Out-of-sample	83.15	87.12	76.92	85.54	79.21	86.32

Table 2: Model Performance across different metrics

In figure 6, we can see that SVM is outperforming both the Logistic Regression and Step-wise Logistic Regression Models based on it's out-of-sample accuracy score. When looking at the position of a sales manager, we know that part of the job description is to evaluate performance and create performance goals. Having a good estimate of future sales help creates performance metrics and evaluate performance actively instead of merely retro-actively based on past data.

We chose to select the best models based on accuracy for the simple reason that both a CH and MM prediction is important in terms of the penalty of being below or under the actual performance. Both being above or under performance (sales numbers wise) can cause unwanted actions to be taken to adjust performance metrics or strategies in the future. While an increase of around 20% accuracy is good over a benchmark, not properly predicting the outcome 17% of the time can drastically alter performance benchmarks due to forecasting errors. To gain a better understanding of where the model is lacking in forecasting performance, we will discuss some performance metrics as shown in appendix D and table 2 for both SVM and Step-Wise GLM due to their out-of-sample accuracy scores being close together.

When comparing both models, it shows that Step-Wise GLM only has one extra wrong prediction over SVM. With Step-Wise GLM more heavily predicting MM wrong over SVM, it can be assumed that the total monetary sales volume of Step-Wise GLM will be too high compared to the actual data and the opposite will be true for SVM. Where more often wrongly predicting CH, the monetary sales volume will be estimated too low. Now since the difference in performance is so small, the sales manager should make a decision as to which error is deemed less punishing. Is understating sales better or overstating sales better? In the end, this is up to the discretion and the unease of the models for the Sales manager but is advised to use SVM to dampen expectations.

Confidence in the SVM model is mediocre since only 83.15% of predictions are correct. This error rate can influence results by a large range dependent on the rowcount of new predictions. As stated under Goal 1, data collection should be continued and expanded and the analysis should be redone in the future when more data is available to increase accuracy scores with time.

References

Glen, S. (2020, July 09). *Variance Inflation Factor*. Retrieved November 25, 2020, from https://www.statisticshowto.com/variance-inflation-factor/

Hastie, T., Friedman, J., & Tisbshirani, R. (2017). *Subset Selection. In The Elements of statistical learning: Data mining, inference, and prediction*. New York: Springer.

Kassambara, A. (2018, March 11). *Logistic Regression Assumptions and Diagnostics in R*. Retrieved November 20, 2020, from http://www.sthda.com/english/articles/36-classification-methods-essentials/148-logistic-regression-assumptions-and-diagnostics-in-r/

Mindrila, D., & Balentyne, P. (n.d.). *Scatterplots and Correlation*. Retrieved November 20, 2020, from https://www.westga.edu/academics/research/vrc/assets/docs/scatterplots_and_correlation_notes.pdf

Sauer, S. (2017, January 24). *Convert logit to probability*. Retrieved November 25, 2020, from https://sebastiansauer.github.io/convert_logit2prob/

Appendix

Appendix A: Codebook

Code Book

Purchase: A factor with levels CH(Citrus Hill) and MM(Minute Maid) indicating which the customer purchased Citrus Hill or Minute Maid Orange Juice (Target Variable)

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 Citrus Hill. Also called List Price for Citrus Hill

PriceMM: Price charged for Minute Maid. Also called List Price for Minute Maid

DiscCH: Discount offered for Citrus Hill

DiscMM: Discount offered for Minute Maid

SpecialCH: Indicator of special on Citrus Hill. Special can be a free gift, loyalty points etc.

SpecialMM: Indicator of special on Minute Maid. Special can be a free gift, loyalty points etc.

LoyalCH: Customer brand loyalty for Citrus Hill. That is, probability to buy Citrus Hill (over Minute Maid) based on prior purchase behavior.

SalePriceMM: Sale price for Minute Maid. This is the difference between the list price and discount.

SalePriceCH: Sale price for Citrus Hill. This is the difference between the list price and discount.

PriceDiff: Sale price of Minute Maid less sale price of Citrus Hill

Store7: A factor with levels No and Yes indicating whether the sale is at Store 7

PctDiscMM: Percentage discount for Minute Maid

PctDiscCH: Percentage discount for Citrus Hill

ListPriceDiff: List price of Minute Maid less list price of Citrus Hill

STORE:(Which of 5 possible stores the sale occurred at

Appendix B: Summary Tables

```
df <- df_backup

print(xtable(summary(df[1:5]),
             caption="Summary of Dataset variable 1 through 5"), comment = FALSE)
```

	Purchase	WeekofPurchase	StoreID	PriceCH	PriceMM
X	CH:653	Min. :227.0	Min. :1.00	Min. :1.690	Min. :1.690
X.1	MM:417	1st Qu.:240.0	1st Qu.:2.00	1st Qu.:1.790	1st Qu.:1.990
X.2		Median :257.0	Median :3.00	Median :1.860	Median :2.090
X.3		Mean :254.4	Mean :3.96	Mean :1.867	Mean :2.085
X.4		3rd Qu.:268.0	3rd Qu.:7.00	3rd Qu.:1.990	3rd Qu.:2.180
X.5		Max. :278.0	Max. :7.00	Max. :2.090	Max. :2.290

Table 3: Summary of Dataset variable 1 through 5

```
print(xtable(summary(df[6:10]),
             caption="Summary of Dataset variable 6 through 10"), comment = FALSE)
```

	DiscCH	DiscMM	SpecialCH	SpecialMM	LoyalCH
X	Min. :0.00000	Min. :0.0000	Min. :0.0000	Min. :0.0000	Min. :0.000011
X.1	1st Qu.:0.00000	1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.:0.325257
X.2	Median :0.00000	Median :0.0000	Median :0.0000	Median :0.0000	Median :0.600000
X.3	Mean :0.05186	Mean :0.1234	Mean :0.1477	Mean :0.1617	Mean :0.565782
X.4	3rd Qu.:0.00000	3rd Qu.:0.2300	3rd Qu.:0.0000	3rd Qu.:0.0000	3rd Qu.:0.850873
X.5	Max. :0.50000	Max. :0.8000	Max. :1.0000	Max. :1.0000	Max. :0.999947

Table 4: Summary of Dataset variable 6 through 10

```
print(xtable(summary(df[11:15]),
             caption="Summary of Dataset variable 11 through 15"), comment = FALSE)
```

	SalePriceMM	SalePriceCH	PriceDiff	Store7	PctDiscMM
X	Min. :1.190	Min. :1.390	Min. :-0.6700	No :714	Min. :0.0000
X.1	1st Qu.:1.690	1st Qu.:1.750	1st Qu.: 0.0000	Yes:356	1st Qu.:0.0000
X.2	Median :2.090	Median :1.860	Median : 0.2300		Median :0.0000
X.3	Mean :1.962	Mean :1.816	Mean : 0.1465		Mean :0.0593
X.4	3rd Qu.:2.130	3rd Qu.:1.890	3rd Qu.: 0.3200		3rd Qu.:0.1127
X.5	Max. :2.290	Max. :2.090	Max. : 0.6400		Max. :0.4020

Table 5: Summary of Dataset variable 11 through 15

```
print(xtable(summary(df[16:18]),
             caption="Summary of Dataset variable 16 through 18"), comment = FALSE)
```

	PctDiscCH	ListPriceDiff	STORE
X	Min. :0.00000	Min. :0.000	Min. :0.000
X.1	1st Qu.:0.00000	1st Qu.:0.140	1st Qu.:0.000
X.2	Median :0.00000	Median :0.240	Median :2.000
X.3	Mean :0.02731	Mean :0.218	Mean :1.631
X.4	3rd Qu.:0.00000	3rd Qu.:0.300	3rd Qu.:3.000
X.5	Max. :0.25269	Max. :0.440	Max. :4.000

Table 6: Summary of Dataset variable 16 through 18

Appendix C: GLM Output

```
summary(glm_model)
```

```
##
## Call:
## glm(formula = train_target ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8320  -0.5371  -0.2247   0.5371   2.8680
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.3565     2.2319   1.504  0.13262
## StoreID.2    -0.1348     0.3196  -0.422  0.67312
## StoreID.3    -0.1623     0.3839  -0.423  0.67236
## StoreID.4    -0.2748     0.4416  -0.622  0.53370
## StoreID.7    -0.8406     0.3334  -2.521  0.01170 *
## PriceCH      -1.3603     2.0721  -0.656  0.51151
## PriceMM       0.2843     1.4376   0.198  0.84325
## LoyalCH      -6.3446     0.4750 -13.358  < 2e-16 ***
## SalePriceMM  -3.3958     1.0765  -3.155  0.00161 **
## SalePriceCH   4.6369     1.4684   3.158  0.00159 **
## Discount.CH   0.3323     0.4293   0.774  0.43883
## Discount.MM  -0.5804     0.4684  -1.239  0.21533
## Special.CH    0.3743     0.3853   0.972  0.33121
## Special.MM    0.3567     0.3306   1.079  0.28071
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1073.9  on 802  degrees of freedom
## Residual deviance:  610.0  on 789  degrees of freedom
## AIC: 638
##
## Number of Fisher Scoring iterations: 5
```

Appendix D: Confusion Matricies for models

Step-Wise GLM

```
confusionMatrix(test_target, as.factor(ifelse(prediction_step_test>0.5,"MM","CH")), positive = "MM")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 151  12
##         MM  34  70
##
##                Accuracy : 0.8277
##                  95% CI : (0.777, 0.871)
##     No Information Rate : 0.6929
##     P-Value [Acc > NIR] : 3.617e-07
##
##                   Kappa : 0.6233
##
##  Mcnemar's Test P-Value : 0.00196
##
##             Sensitivity : 0.8537
##             Specificity : 0.8162
##          Pos Pred Value : 0.6731
##          Neg Pred Value : 0.9264
##              Prevalence : 0.3071
##          Detection Rate : 0.2622
##    Detection Prevalence : 0.3895
##       Balanced Accuracy : 0.8349
##
##        'Positive' Class : MM
##
```

SVM

```
confusionMatrix(test_target, prediction_svm_test, positive = "MM")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 142  21
##         MM  24  80
##
##                Accuracy : 0.8315
##                  95% CI : (0.7811, 0.8743)
##     No Information Rate : 0.6217
##     P-Value [Acc > NIR] : 5.746e-14
##
##                   Kappa : 0.6438
##
##  Mcnemar's Test P-Value : 0.7656
```

```
##
##             Sensitivity : 0.7921
##             Specificity : 0.8554
##          Pos Pred Value : 0.7692
##          Neg Pred Value : 0.8712
##              Prevalence : 0.3783
##          Detection Rate : 0.2996
##    Detection Prevalence : 0.3895
##       Balanced Accuracy : 0.8238
##
##        'Positive' Class : MM
##
```