Predicting Citrus Hill vs Minute Maid Orange Juice Purchases Based on Sales Information

Data Analysis Assignment Lesson 10

David Chen

April 7th, 2020

Introduction

For our analysis, we will classify/predict purchases of Citrus Hill (CH) vs Minute Maid (MM) brands of orange juice based on pricing and store characteristics contained within the `OJ` dataset. We will fit a support vector classifier, a support vector machine (SVM) with a radial kernel, and a support vector machine with a polynomial kernel, applying varying values for the cost parameter. After judging model performance based on training and testing error rates, we found that the SVM with a polynomial kernel and cost = 1 performed the best.

Data

The `OJ` dataset is sourced from the ISLR package, originally published in 1998. This dataset contains 1070 observations, where customers purchased either Citrus Hill or Minute Maid orange juice. The variables predominately consist of store/pricing information. The following variable descriptions are copied from the CRAN documentation:

- `Purchase` - A factor with levels CH and MM indicating whether the customer purchased Citrus Hill or Minute Maid Orange Juice
- `WeekofPurchase` - Week of purchase
- `StoreID` - Store ID
- `PriceCH` - Price charged for CH
- `PriceMM` - Price charged for MM
- `DiscCH` - Discount offered for CH
- `DiscMM` - Discount offered for MM
- `SpecialCH` - Indicator of special on CH
- `SpecialMM` - Indicator of special on MM
- `LoyalCH` - Customer brand loyalty for CH
- `SalePriceMM` - Sale price for MM
- `SalePriceCH` - Sale price for CH
- `PriceDiff` - Sale price of MM less sale price of CH
- `Store7` - A factor with levels No and Yes indicating whether the sale is at Store 7
- `PctDiscMM` - Percentage discount for MM
- `PctDiscCH` - Percentage discount for CH
- `ListPriceDiff` - List price of MM less list price of CH
- `STORE` - Which of 5 possible stores the sale occured at

Exploratory Analysis

From the list of variables, we immediately note that many of the variables convey the same information as previous ones. Thus, we note that there are percent and total discount variables, and multiple store ID variables. Thus, we investigate further to see if we can remove any variables.

First, we examine the relationship between `PctDisc` and `Disc` variables. As shown in Figure 1, there is a near perfect correlation, and thus we will remove `PctDiscMM` and `PctDiscCH` (note that the sales prices are relatively close, so converting to percents is less meaningful).

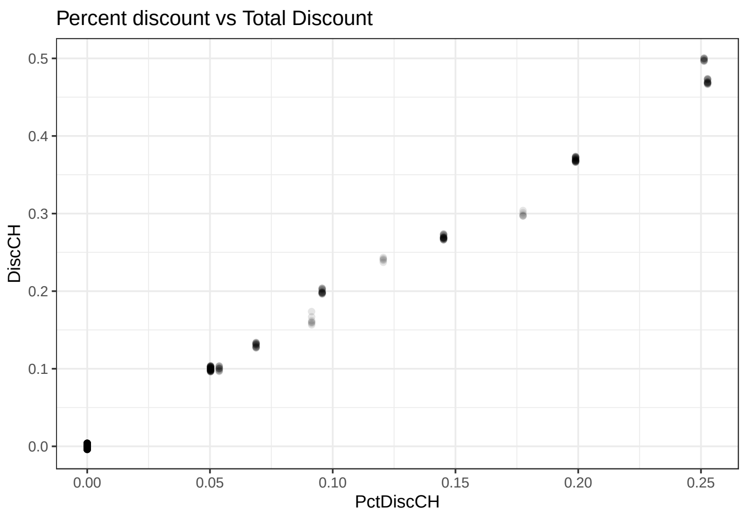

Figure 1: Percent discount vs Total Discount

Next, we examine the relationship between Stores and Purchases (Figure 2). Since we observe that all the purchases seem evenly distributed within a store besides Store 7, we keep the `Store7` variable. We will remove `StoreID` and `STORE`, as we are less interested in using the behaviors of the other stores to predict purchases.

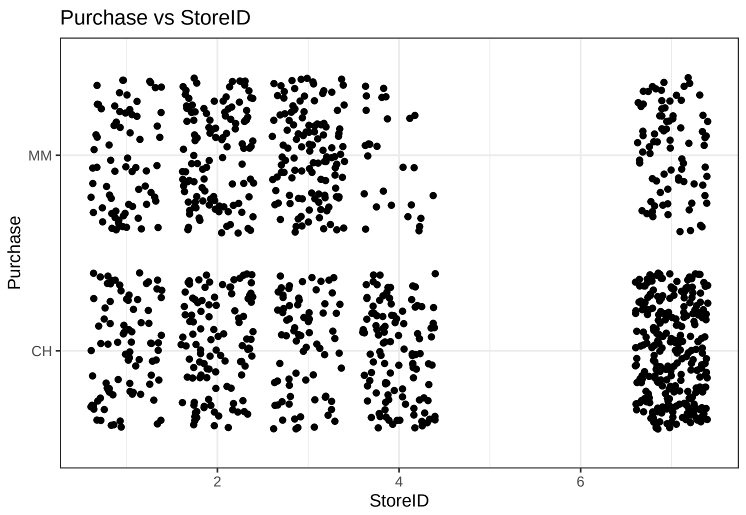

Figure 2: Purchase vs StoreID

Lastly, we examine the relationship between `WeekofPurchase` and `Purchase`. While Figure 3 seems to suggest that as the weeks go on, the purchases of CH increases, there is a great deal of uncertainty as to why this trend exists. Since we are unclear about external factors and whether or not there is a cyclical relationship (i.e. if we go further into time the purchases may decrease), we will remove this variable for our analysis.

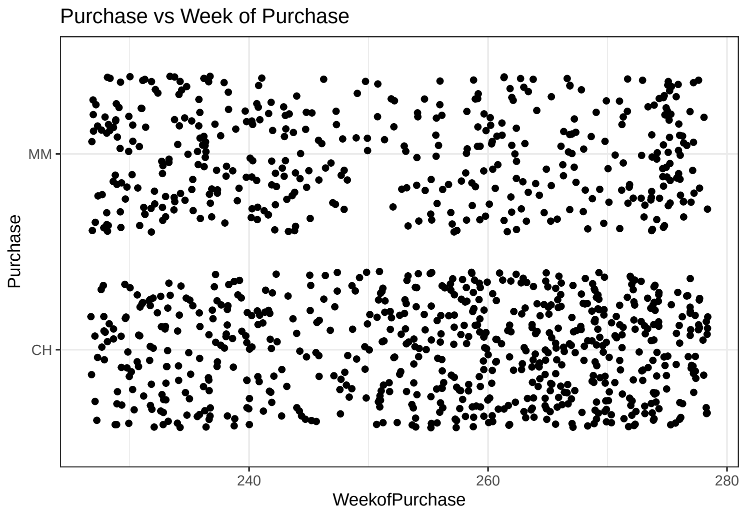

Figure 3: Purchase vs Week of Purchase

EDA

Lastly, for the sake of reference, we review a summary of all the remaining `OJ` variables. We note that first of all, there are more purchases of CH than of MM. Additionally, we note that the price of MM is a little bit higher, but not substantially. We can also note that it seems that `DiscCH` is a fixed value (the max of 0.5), while `DiscMM` has more of a range. Lastly, we observe that Store 7 appears 356 times, while the other 4 stores appear 714 times.

```
##  Purchase    PriceCH        PriceMM         DiscCH     
##  CH:653   Min.   :1.69   Min.   :1.69   Min.   :0.000  
##  MM:417   1st Qu.:1.79   1st Qu.:1.99   1st Qu.:0.000  
##           Median :1.86   Median :2.09   Median :0.000  
##           Mean   :1.87   Mean   :2.08   Mean   :0.052  
##           3rd Qu.:1.99   3rd Qu.:2.18   3rd Qu.:0.000  
##           Max.   :2.09   Max.   :2.29   Max.   :0.500  
##      DiscMM        SpecialCH       SpecialMM        LoyalCH     
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000  
##  1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.325  
##  Median :0.000   Median :0.000   Median :0.000   Median :0.600  
##  Mean   :0.123   Mean   :0.148   Mean   :0.162   Mean   :0.566  
##  3rd Qu.:0.230   3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:0.851  
##  Max.   :0.800   Max.   :1.000   Max.   :1.000   Max.   :1.000  
##   SalePriceMM    SalePriceCH     PriceDiff      Store7   
##  Min.   :1.19   Min.   :1.39   Min.   :-0.670   No :714  
##  1st Qu.:1.69   1st Qu.:1.75   1st Qu.: 0.000   Yes:356  
##  Median :2.09   Median :1.86   Median : 0.230            
##  Mean   :1.96   Mean   :1.82   Mean   : 0.146            
##  3rd Qu.:2.13   3rd Qu.:1.89   3rd Qu.: 0.320            
##  Max.   :2.29   Max.   :2.09   Max.   : 0.640            
##  ListPriceDiff
##  Min.   :0.000
##  1st Qu.:0.140
##  Median :0.240
##  Mean   :0.218
##  3rd Qu.:0.300
##  Max.   :0.440
```

Analysis

For this analysis, we will split the data into training and testing sets. From the original sample of 1070 customer purchases, we will randomly split 800 (74.8%) observations into the training set, while the remaining 270 observations become the testing set.

Support Vector Classifier (Linear Kernel)

First, we attempt to fit a support vector classifier (linear kernel) to the training data using cost = 0.01. We will attempt to predict the brand purchased using all the remaning variables. The predictors will be scaled to avoid variables such as cost from overshadowing inherently smaller variables such as discount amount.

After fitting the model, we observe that there was a total of 452 support vectors. This indicates that out of the 800 total observations in the training set, the 452 directly influence the boundary hyperplane. If any of those 452 observations changed, then the boundary boundary would have as well.

```
##
## Call:
## svm(formula = Purchase ~ ., data = training, kernel = "linear",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  452
##
##  ( 226 226 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

To determine the performance of the support vector classifier, we examine the training and testing error rates. As observed from the two tables below, this classifier has a training error rate of 0.17 and a testing error rate of 0.159. While these seem alright, we need different models to compare these rates to. Thus, we proceed forward with different costs.

Table 1: Testing Data with Linear Kernel (cost = 0.01)

	True CH	True MM
CH	440	80
MM	56	224

Table 2: Testing Data with Linear Kernel (cost = 0.01)

	True CH	True MM
CH	139	25
MM	18	88

Changing cost (Linear Kernel)

While previously we considered cost = 0.01, we want to check if varying cost values might improve the model performance. After testing values between 0.01 and 10, we find that surprisingly, cost = 0.01 resulted in the lowest training error rate. Thus, the results are the exact same as previously shown.

```
##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = training,
##     ranges = list(cost = c(0.01, 0.1, 1, 5, 10)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  452
##
##  ( 226 226 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

Support Vector Machine (Radial Kernel)

Next, we want to check if a support vector machine with a radial kernel can perform better than the linear classifier. Compared to the support vector classifier, this SVM would allow the decision boundary to be radial as opposed to linear.

First, we observe the model with cost = 0.01. From the summary output, we first note that the number of support vectors increases to 613, much more than the previous model.

```
##
## Call:
## svm(formula = Purchase ~ ., data = training, kernel = "radial",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.01
##
## Number of Support Vectors:  613
##
##  ( 309 304 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

Checking for the training and testing error rates, we observe values of 0.38 and 0.419 respectively. These are values over double the support vector classifiers previously, seemingly indicating that a radial kernel may not be as appropriate. However, to be sure, we check multiple cost values to see if the error rates decrease.

Table 3: Training Data with Radial Kernel (cost = 0.01)

	True CH	True MM
CH	496	304
MM	0	0

Table 4: Testing Data with Radial Kernel (cost = 0.01)

	True CH	True MM
CH	157	113
MM	0	0

Changing cost (Radial Kernel)

Post-submission note: We should also explore different values of gamma here. With only the default value, we risk missing the optimal model.

Checking for the training error rates across the a range of cost values (0.01-10), we observe that the best model has a cost value of 1, with 379 support vectors. Checking for the testing and training error rates, we observe values of 0.165 and 0.152 respectively, making the rates even better than the support vector classifier's rates.

```
##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = training,
##     ranges = list(cost = c(0.01, 0.1, 1, 5, 10)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  379
##
##  ( 191 188 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

Support Vector Machine (Polynomial Kernel)

Lastly, we consider a support vector machine with a polynomial kernel and degree = 2.

Just like previous models, we will first begin with cost = 0.01. From the summary statistics, we observe 614 support vectors, similar to the radial kernel.

```
##
## Call:
## svm(formula = Purchase ~ ., data = training, kernel = "polynomial",
##     degree = 2, cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.01
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  614
##
##  ( 310 304 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

While the training error rate of 0.38 and testing error rate of 0.419 seem to suggest simiar behavior to the initial radial model, the fact that all purchases were classified as CH leads us to doubt this model.

Table 5: Training Data with Polynomial Kernel (cost = 0.01)

	True CH	True MM
CH	496	304
MM	0	0

Table 6: Testing Data with Polynomial Kernel (cost = 0.01)

	True CH	True MM
CH	157	113
MM	0	0

Changing cost (Polynomial Kernel)

Checking the range of cost values, we find that cost = 2 resulted in the lowest training error rate, so we proceed forward with that model, containing 353 support vectors. While the training and testing error rates decrease to 0.169 and 0.178 respectively, the testing error rate is still higher than both the linear and radial kernels.

```
##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = training,
##     ranges = list(cost = c(0.01, 0.1, 1, 5, 10)), kernel = "polynomial",
##     degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  353
##
##  ( 177 176 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

Table 7: Training Data with Polynomial Kernel (cost = 2)

	True CH	True MM
CH	448	87
MM	48	217

Table 8: Testing Data with Polynomial Kernel (cost = 2)

	True CH	True MM
CH	137	28
MM	20	85

Comparison

Comparing all the models together, we find that the support vector machine with a radial kernel and cost = 1 was close and had a better training error rate.

Table 9: Model Training and Testing Error Rates

	training_rates	testing_rates
Linear	0.170	0.159
Best Linear	0.170	0.159
Radial	0.380	0.419
Best Radial	0.165	0.152
Poly	0.380	0.419
Best Poly	0.169	0.178

Conclusion

After removing redundant variables and `WeekofPurchase`, we found that after conducting a support vector classifier, a support vector machine with a radial and a polynomial kernel (with degree = 2), the model with the radial kernel performed the best in terms of testing and training error rates. With a training error rate of 0.165, and a testing error rate of 0.152, that would likely be the best model to use moving forward.