IST-565 Final Project: Black Friday Market Basket Analysis and Consumer Predictions

Joseph Hartman

September 22, 2018

## Abstract

Improving customer insight is a goal that all retail companies strive to accomplish. With the digital age and the surplus of consumer information these companies are aggregating, it can difficult finding these insights hidden in the sea of data. Fortunately, using data mining techniques one can derive some insight about consumer buying habits and target markets when creating their marketing material. To show an example of the application of data mining in this sphere, we’ll be using a sample (10,000 observations) of the Black Friday dataset found on [Kaggle](https://www.kaggle.com/mehdidag/black-friday). We’ll be trying to answer two questions and use two different Data Mining techniques to answer them:

1. What kinds of products do consumers buy in conjunction with other products? (Association Rule)
2. Can we create a method of predicting what items our consumers will buy and how much they’ll spend based on demographic information? (Naive Bayes & Random Forest)

## Data Exploration

Packages that are used in this project

library(caret)

library(psych)

library(RWeka)  
library(randomForest)

library(arulesViz)

library(arules)

First step is to import the data and remove any null values that might mess with our models.

bf.data <- na.omit(read.csv("~/data/BlackFriday.csv"))

The read.csv function usually does a decent job at classify variables in the right types, but sometimes, additional work is needed to convert certain variables into factors that were imported in as a numeric. We also don’t need the first two columns which are just Customer and Order ID numbers.

#removes customer and order ID numbers  
bf.data <- bf.data[,-(1:2)]  
#changes everything but Purchase to nominal variables  
bf.data$Occupation <- factor(bf.data$Occupation)  
bf.data$Marital\_Status <- factor(ifelse(bf.data$Marital\_Status == TRUE, "yes", "no"))  
bf.data$Product\_Category\_1 <- factor(bf.data$Product\_Category\_1)  
bf.data$Product\_Category\_2 <- factor(bf.data$Product\_Category\_2)  
bf.data$Product\_Category\_3 <- factor(bf.data$Product\_Category\_3)

Now for the purposes of this project and just to reduce the computation time for these models we’ll only be using 10,000 observations from this dataset in order to demonstrate certain data mining techniques while still having enough data to generate decent models.

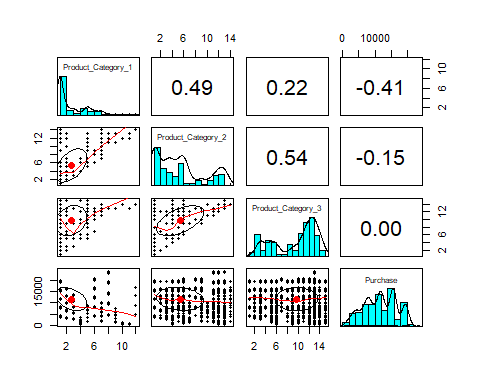
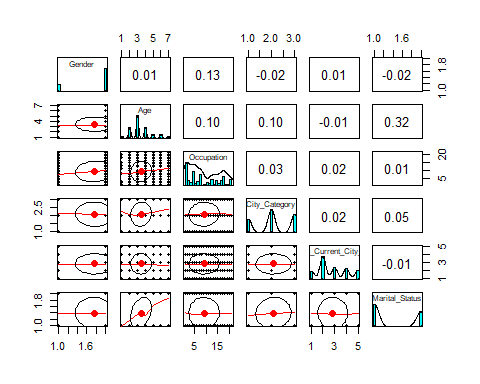
set.seed(1138)  
sample.index <- sample(1:nrow(bf.data), 10000, replace = FALSE)  
bf.data <- bf.data[sample.index,]  
knitr::kable(summary(bf.data))

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Gender | Age | Occupation | City\_Category | Stay\_In\_Current\_City\_Years | Marital\_Status | Product\_Category\_1 | Product\_Category\_2 | Product\_Category\_3 | Purchase |
|  | F:2271 | 0-17 : 280 | 4 :1329 | A:2473 | 0 :1352 | no :6025 | 1 :5452 | 2 :2642 | 16 :1893 | Min. : 375 |
|  | M:7729 | 18-25:1880 | 0 :1248 | B:4177 | 1 :3543 | yes:3975 | 5 :1107 | 8 :1628 | 15 :1659 | 1st Qu.: 7881 |
|  | NA | 26-35:4036 | 7 :1031 | C:3350 | 2 :1863 | NA | 2 : 953 | 4 :1233 | 14 :1121 | Median :11768 |
|  | NA | 36-45:2027 | 17 : 859 | NA | 3 :1753 | NA | 3 : 808 | 5 : 957 | 5 :1070 | Mean :11674 |
|  | NA | 46-50: 762 | 1 : 847 | NA | 4+:1489 | NA | 6 : 521 | 15 : 795 | 17 :1008 | 3rd Qu.:15655 |
|  | NA | 51-55: 686 | 12 : 593 | NA | NA | NA | 8 : 518 | 14 : 721 | 8 : 771 | Max. :23939 |
|  | NA | 55+ : 329 | (Other):4093 | NA | NA | NA | (Other): 641 | (Other):2024 | (Other):2478 | NA |

The psych package has an excellent visualization function called pairs.panels that combines distributions, correlation values, and correlation scatterplots into one visualization. Using this, we can see how various parts of our customer data correlate to one another and see how our products correlate with one another and the basket purchase cost, which will be useful going into our association rules mining.

pairs.panels(bf.data[,1:6])

pairs.panels(bf.data[,7:10])



From here we can see that the factors that describe our customers (age, gender, marital status, etc.) are probably not correlated except age and marital status which have a slight correlation. On the other hand, it seems that the product categories have some degree of correlation. This makes sense as these our essentially just items and that the numbers describe the same product categories amongst all three product category columns (i.e product category, item 12 is the same in Product\_Category\_1 as it is in Product\_Category\_3). It seems the purchase value has some negative correlation amongst the first product category column but not so much in the second and none in the third.

Another interesting point to note is that there is a good sign of correlation between Product\_Category\_1 and Product\_Category\_2 (0.49) and Product\_Category\_2 and Product\_Category\_3. From here we can infer a likelihood that purchasing an item in 1 might mean purchasing an item in 2 and that purchasing an item might lead to purchasing an item in 3.

#### Creating the training and testing dataset

To verify our predictive models, we’ll be using the hold out method for final verification and cross-validation in between

set.seed(1138) #sets a seed for reproducability  
index <- createDataPartition(bf.data$Product\_Category\_1, p = 0.8, list = FALSE) #creates an index in which to split the training dataset  
bf.test <- bf.data[-index,] #creates a dataset for model validation, 0.2 of the original training dataset  
bf.train <- bf.data[index,] #creates a dataset for training, .8 of the original dataset

## Business Question 1: Market Basket Analysis

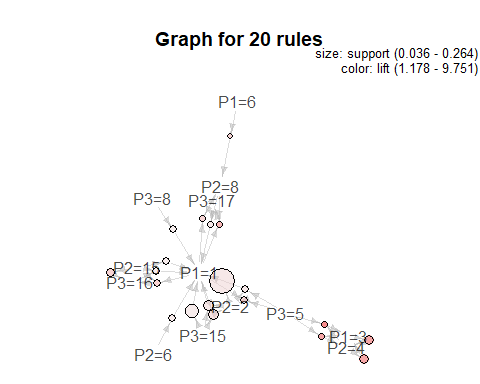
One task we can accomplish with this data set is seeing what types of items are customers buy in conjunction with one another. This is a task well suited for Association Rule (AR) mining.

#### AR Mining

names(bf.data)[7:9] <- c("P1", "P2", "P3") #renames Product Category to something more visualization friendly  
rules <- apriori(bf.data[,7:9], parameter = list(supp = 0.01, conf = 0.6, maxlen = 4), control = list(verbose = FALSE)) #creates the rules  
rules\_conf <- sort(rules, by="support", decreasing=TRUE) #brings 'high-support' rules to the top  
inspect(head(rules\_conf, 20)) # show the support, lift and confidence for all rules

## lhs rhs support confidence lift count  
## [1] {P2=2} => {P1=1} 0.2642 1.0000000 1.834189 2642   
## [2] {P3=15} => {P1=1} 0.1278 0.7703436 1.412956 1278   
## [3] {P2=2,P3=15} => {P1=1} 0.0893 1.0000000 1.834189 893   
## [4] {P1=1,P3=15} => {P2=2} 0.0893 0.6987480 2.644769 893   
## [5] {P1=3} => {P2=4} 0.0789 0.9764851 7.919588 789   
## [6] {P2=4} => {P1=3} 0.0789 0.6399027 7.919588 789   
## [7] {P2=15} => {P3=16} 0.0672 0.8452830 4.465309 672   
## [8] {P2=15} => {P1=1} 0.0584 0.7345912 1.347379 584   
## [9] {P2=6} => {P1=1} 0.0535 0.7664756 1.405861 535   
## [10] {P2=2,P3=5} => {P1=1} 0.0509 1.0000000 1.834189 509   
## [11] {P1=1,P3=5} => {P2=2} 0.0509 1.0000000 3.785011 509   
## [12] {P3=8} => {P1=1} 0.0495 0.6420233 1.177592 495   
## [13] {P2=15,P3=16} => {P1=1} 0.0491 0.7306548 1.340159 491   
## [14] {P1=1,P2=15} => {P3=16} 0.0491 0.8407534 4.441381 491   
## [15] {P1=3,P3=5} => {P2=4} 0.0442 1.0000000 8.110300 442   
## [16] {P2=4,P3=5} => {P1=3} 0.0442 0.7878788 9.750975 442   
## [17] {P2=8,P3=17} => {P1=1} 0.0399 0.8599138 1.577245 399   
## [18] {P1=1,P3=17} => {P2=8} 0.0399 0.7061947 4.337805 399   
## [19] {P1=1,P2=8} => {P3=17} 0.0399 0.6808874 6.754835 399   
## [20] {P1=6} => {P2=8} 0.0360 0.6909789 4.244342 360

plot(head(rules\_conf, 20), method = "graph") #displays a rules web

From here we see that types of items represented by 1 and 2 appear quite frequently in the top twenty rules. These items are probably common items or popular items that people buy during Black Friday. It also seems that type 1 items are bought in conjunction with various other items. In other words, despite whatever a person buys, there’s a good chance they might also buy items of type 1. We also see that the support values drop sharply going down the list, despite consistently high values in confidence and list. This may be due that number of combinations of a three item basket is quite large and so only a few common basket trends would appear.

## Business Question 2: Consumer Prediction

We’re going to use our models for two different predictions:

* Based of our customer’s information, we’ll predict what kind of products they will buy
* Based of our customer’s information, we’ll predict how much they will spend

Since most of our data is nominal and ordinal, we’ll be using Naive Bayes and Random Forest models, both well suited for predicting nominal outputs when given nominal predictors.

We’ll be creating 4 different models for each training method (1 for each product category and 1 for overall purchase cost) for a total of 8 models, using the first 6 columns as our predictors.

#### Naive Bayes

#discretizes numeric purchase variable to be used as outcome variable in Naive Bayes  
purchase.disc <- discretize(bf.data$Purchase, method = "cluster", breaks = 5)  
purchase.disc.train <- purchase.disc[index]  
purchase.disc.test <- purchase.disc[-index]  
  
NB <- make\_Weka\_classifier("weka/classifiers/bayes/NaiveBayes") #determines classifier  
#creates naive bayes models  
nb1 <- NB(Product\_Category\_1 ~ Gender + Age + Occupation + City\_Category + Stay\_In\_Current\_City\_Years + Marital\_Status, data = bf.train)  
nb2 <- NB(Product\_Category\_2 ~ Gender + Age + Occupation + City\_Category + Stay\_In\_Current\_City\_Years + Marital\_Status, data = bf.train)  
nb3 <- NB(Product\_Category\_3 ~ Gender + Age + Occupation + City\_Category + Stay\_In\_Current\_City\_Years + Marital\_Status, data = bf.train)  
nb4 <- NB(purchase.disc.train ~ Gender + Age + Occupation + City\_Category + Stay\_In\_Current\_City\_Years + Marital\_Status, data = bf.train)

Now that we have our models we can predict the outcomes of our testing dataset, this method of model validation is using the hold-out method. We’ll build confusion matrices for each of the outcome variables and then plot their overall accuracy side by side.

#creates predictions  
nb.predict1 <- predict(nb1, newdata = bf.test[,1:6])  
nb.predict2 <- predict(nb2, newdata = bf.test[,1:6])  
nb.predict3 <- predict(nb3, newdata = bf.test[,1:6])  
nb.predict4 <- predict(nb4, newdata = bf.test[,1:6])  
  
#creates confusion matricies  
nb.cm1 <- confusionMatrix(nb.predict1, bf.test$Product\_Category\_1)  
nb.cm2 <- confusionMatrix(nb.predict2, bf.test$Product\_Category\_2)  
nb.cm3 <- confusionMatrix(nb.predict3, bf.test$Product\_Category\_3)  
nb.cm4 <- confusionMatrix(nb.predict4, purchase.disc.test)  
  
#displays confusion matricies  
nb.cm1$table

## Reference  
## Prediction 1 2 3 4 5 6 8 10 11 12 13 15  
## 1 1083 188 158 65 221 104 103 21 24 2 13 1  
## 2 0 0 0 0 0 0 0 0 0 0 0 0  
## 3 7 1 3 0 0 0 0 0 1 0 0 0  
## 4 0 0 0 0 0 0 0 0 0 0 0 0  
## 5 0 1 0 0 0 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0 0 0 0 0 0  
## 8 0 0 0 0 0 0 0 0 0 0 0 0  
## 10 0 0 0 0 0 0 0 0 0 0 0 0  
## 11 0 0 0 0 0 0 0 0 0 0 0 0  
## 12 0 0 0 0 0 0 0 0 0 0 0 0  
## 13 0 0 0 0 0 0 0 0 0 0 0 0  
## 15 0 0 0 0 0 0 0 0 0 0 0 0

nb.cm2$table

## Reference  
## Prediction 2 3 4 5 6 8 9 10 11 12 13 14 15 16  
## 2 489 27 224 166 113 279 29 13 76 9 59 120 132 11  
## 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 4 6 1 10 7 5 8 1 1 3 0 6 5 6 0  
## 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 8 34 1 20 19 15 49 4 6 4 2 9 17 10 0  
## 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 13 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 14 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0

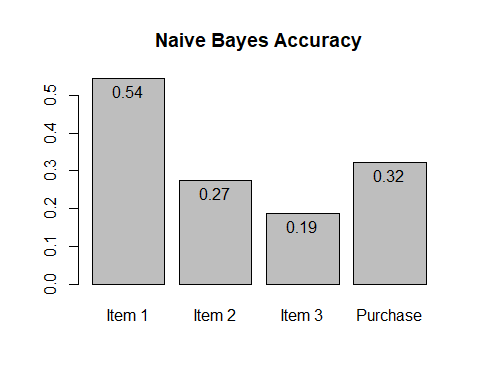
nb.cm3$table

## Reference  
## Prediction 3 4 5 6 8 9 10 11 12 13 14 15 16 17 18  
## 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 5 1 1 25 4 5 9 1 0 4 2 8 13 16 10 1  
## 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 8 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0  
## 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 13 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 14 1 0 19 2 9 8 1 2 7 5 28 20 34 17 3  
## 15 1 2 57 16 40 44 1 6 24 16 55 99 113 53 20  
## 16 4 17 113 28 81 77 10 9 62 40 120 208 211 105 31  
## 17 0 1 3 1 14 6 1 0 3 4 8 14 8 12 1  
## 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

nb.cm4$table

## Reference  
## Prediction [375,8.5e+03) [8.5e+03,1.34e+04) [1.34e+04,1.75e+04)  
## [375,8.5e+03) 411 364 281  
## [8.5e+03,1.34e+04) 181 198 145  
## [1.34e+04,1.75e+04) 42 39 34  
## [1.75e+04,2.15e+04) 0 0 0  
## [2.15e+04,2.39e+04] 0 0 0  
## Reference  
## Prediction [1.75e+04,2.15e+04) [2.15e+04,2.39e+04]  
## [375,8.5e+03) 166 7  
## [8.5e+03,1.34e+04) 105 2  
## [1.34e+04,1.75e+04) 21 0  
## [1.75e+04,2.15e+04) 0 0  
## [2.15e+04,2.39e+04] 0 0

#plots overall hold-out method accuracy  
nb.accuracy <- c(nb.cm1$overall[1], nb.cm2$overall[1], nb.cm3$overall[1], nb.cm4$overall[1])  
nb.barplot <- barplot(nb.accuracy, main = "Naive Bayes Accuracy", names.arg = c("Item 1", "Item 2", "Item 3", "Purchase"))  
text(nb.barplot, nb.accuracy, labels=round(nb.accuracy,2), pos=3, offset=-1)



From here, we can see due to the unbalanced nature of the product information that our Naive Bayes model doesn’t do a great job at predicting based of our customer data as the models are heavily biased. We identified earlier from our market basket analysis that Product\_Category\_1 = 1 occurs many times. Because this item appears a lot in our dataset in relation to other items, it’s throwing off our results.

#### Random Forest

Let’s see if the Random Forest method can create a better prediction

set.seed(1138)  
#Builds the random forests with 5 fold CV   
bf.train$purchase.disc.train <- purchase.disc.train  
rf1 <- train(Product\_Category\_1 ~ Gender + Age + Occupation + City\_Category + Stay\_In\_Current\_City\_Years + Marital\_Status,   
 data = bf.train, method = "rf",  
 tuneGrid = data.frame(mtry = seq(1:5)),  
 trControl = trainControl(method = "cv", number = 5)) #5   
rf2 <- train(Product\_Category\_2 ~ Gender + Age + Occupation + City\_Category + Stay\_In\_Current\_City\_Years + Marital\_Status,   
 data = bf.train, method = "rf",  
 tuneGrid = data.frame(mtry = seq(1:5)),  
 trControl = trainControl(method = "cv", number = 5))  
rf3 <- train(Product\_Category\_3 ~ Gender + Age + Occupation + City\_Category + Stay\_In\_Current\_City\_Years + Marital\_Status,   
 data = bf.train, method = "rf",  
 tuneGrid = data.frame(mtry = seq(1:5)),  
 trControl = trainControl(method = "cv", number = 5))  
rf4 <- train(purchase.disc.train ~ Gender + Age + Occupation + City\_Category + Stay\_In\_Current\_City\_Years + Marital\_Status,   
 data = bf.train, method = "rf",  
 tuneGrid = data.frame(mtry = seq(1:5)),  
 trControl = trainControl(method = "cv", number = 5))

Again, using the hold-out method, we’ll verify our models by building a confusion matrix and then verify our models’ overall accuracy side by side.

#creates predictions  
rf.predict1 <- predict(rf1, newdata = bf.test[,1:6])  
rf.predict2 <- predict(rf2, newdata = bf.test[,1:6])  
rf.predict3 <- predict(rf3, newdata = bf.test[,1:6])  
rf.predict4 <- predict(rf4, newdata = bf.test[,1:6])  
#creates confusion matricies  
rf.cm1 <- confusionMatrix(rf.predict1, bf.test$Product\_Category\_1)  
rf.cm2 <- confusionMatrix(rf.predict2, bf.test$Product\_Category\_2)  
rf.cm3 <- confusionMatrix(rf.predict3, bf.test$Product\_Category\_3)  
rf.cm4 <- confusionMatrix(rf.predict4, purchase.disc.test)  
  
#displays confusion matricies  
rf.cm1$table

## Reference  
## Prediction 1 2 3 4 5 6 8 10 11 12 13 15  
## 1 1090 190 161 65 221 104 103 21 25 2 13 1  
## 2 0 0 0 0 0 0 0 0 0 0 0 0  
## 3 0 0 0 0 0 0 0 0 0 0 0 0  
## 4 0 0 0 0 0 0 0 0 0 0 0 0  
## 5 0 0 0 0 0 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0 0 0 0 0 0  
## 8 0 0 0 0 0 0 0 0 0 0 0 0  
## 10 0 0 0 0 0 0 0 0 0 0 0 0  
## 11 0 0 0 0 0 0 0 0 0 0 0 0  
## 12 0 0 0 0 0 0 0 0 0 0 0 0  
## 13 0 0 0 0 0 0 0 0 0 0 0 0  
## 15 0 0 0 0 0 0 0 0 0 0 0 0

rf.cm2$table

## Reference  
## Prediction 2 3 4 5 6 8 9 10 11 12 13 14 15 16  
## 2 529 29 254 191 133 336 34 20 83 11 74 142 148 11  
## 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 8 0 0 0 1 0 0 0 0 0 0 0 0 0 0  
## 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 13 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 14 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0

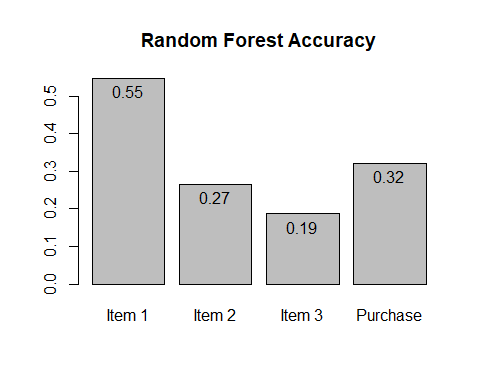
rf.cm3$table

## Reference  
## Prediction 3 4 5 6 8 9 10 11 12 13 14 15 16 17 18  
## 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 5 0 1 15 2 3 6 0 0 3 1 4 9 9 4 0  
## 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 8 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0  
## 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 13 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## 14 0 0 6 0 2 6 0 2 5 1 10 7 9 3 0  
## 15 1 1 32 10 25 27 2 6 15 11 36 61 82 31 16  
## 16 6 18 164 39 118 105 12 9 76 53 169 273 281 154 40  
## 17 0 1 0 0 1 0 0 0 1 1 0 4 1 5 0  
## 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

rf.cm4$table

## Reference  
## Prediction [375,8.5e+03) [8.5e+03,1.34e+04) [1.34e+04,1.75e+04)  
## [375,8.5e+03) 583 544 413  
## [8.5e+03,1.34e+04) 51 56 46  
## [1.34e+04,1.75e+04) 0 1 1  
## [1.75e+04,2.15e+04) 0 0 0  
## [2.15e+04,2.39e+04] 0 0 0  
## Reference  
## Prediction [1.75e+04,2.15e+04) [2.15e+04,2.39e+04]  
## [375,8.5e+03) 255 9  
## [8.5e+03,1.34e+04) 37 0  
## [1.34e+04,1.75e+04) 0 0  
## [1.75e+04,2.15e+04) 0 0  
## [2.15e+04,2.39e+04] 0 0

#plots overall hold-out method accuracy  
rf.accuracy <- c(rf.cm1$overall[1], rf.cm2$overall[1], rf.cm3$overall[1], rf.cm4$overall[1])  
rf.barplot <- barplot(rf.accuracy, main = "Random Forest Accuracy", names.arg = c("Item 1", "Item 2", "Item 3", "Purchase"))  
text(rf.barplot, rf.accuracy, labels=round(rf.accuracy,2), pos=3, offset=-1)



Again, due to the unbalanced nature of our data set our models are still very biased (if not more biased then our Naive Bayes models). We do see a 1% increase in accuracy when predicting items in Product\_Category\_1 but overall, our accuracy pretty much remains the same.

## 

## Conclusion

There are a multitude of factors as to why our models did not perform to the level we expected, including:

* Heavily unbalanced outcome variables
* lack of lengthy tuning processes (i.e. sequencing over a greater range of values for mtry and ntree to find ideal parameters)
* sampling 10,000 observations wasn’t enough

Fixing problems 2 and 3 are easy to implement but come at the cost of increased computation time. As for problem 1, one possible solution might be to implement boosting (using the adaboost or gbm packages), an ensemble learning method that weighs unlikely outcomes more than likely outcomes over successive rounds of training. This might help reduce the bias problem and is something to look into in the future to increase model performance.

## Citations

Michael Hahsler (2018). arulesViz: Visualizing Association Rules and Frequent Itemsets. R package version 1.3-1. https://CRAN.R-project.org/package=arulesViz

Michael Hahsler, Christian Buchta, Bettina Gruen and Kurt Hornik (2018). arules: Mining Association Rules and Frequent Itemsets. R package version 1.6-1. https://CRAN.R-project.org/package=arules

Michael Hahsler, Bettina Gruen and Kurt Hornik (2005), arules - A Computational Environment for Mining Association Rules and Frequent Item Sets. Journal of Statistical Software 14/15. URL: http://dx.doi.org/10.18637/jss.v014.i15.

Michael Hahsler, Sudheer Chelluboina, Kurt Hornik, and Christian Buchta (2011), The arules R-package ecosystem: Analyzing interesting patterns from large transaction datasets. Journal of Machine Learning Research, 12:1977--1981. URL: http://jmlr.csail.mit.edu/papers/v12/hahsler11a.html.

Hornik K, Buchta C, Zeileis A (2009). “Open-Source Machine Learning: R Meets Weka.” \_Computational Statistics\_, \*24\*(2), 225-232. doi:10.1007/s00180-008-0119-7 (URL: http://doi.org/10.1007/s00180-008-0119-7).

Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan and Tyler Hunt. (2018). caret: Classification and Regression Training. R package version 6.0-80. https://CRAN.R-project.org/package=caret

A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18--22.

Revelle, W. (2018) psych: Procedures for Personality and Psychological Research, Northwestern University, Evanston, Illinois, USA, https://CRAN.R-project.org/package=psych Version = 1.8.4.

Witten IH, Frank E (2005). \_Data Mining: Practical Machine Learning Tools and Techniques\_, 2nd edition. Morgan Kaufmann, San Francisco.