4.3: Plan of Attack

jueves, 16 de enero de 2020 19:28

INTRODUCTION

Your Task

You have been asked by Michael Ortiz, VP of Alert Analytics, to investigate predictive models using machine learning methods. You will apply these models to your Large Matrix file to complete the analysis of overall sentiment toward both iPhone and Samsung Galaxy. In this task you will identify one optimized model to predict the overall sentiment toward iPhones and one optimized model to predict overall sentiment toward Samsung Galaxy handsets. You must use R for your analysis.

This task requires you to prepare two deliverables:

1. **Summary of Findings:** a client-ready report of no more than five pages in Word that includes:
   - Your narrative of the data supported by the results.
   - How confident you are in the results. This should contain three parts:
     - The reported performance metrics from R
     - Your personal sense of how well the attributes you are measuring will actually capture pages that have relevant sentiment.
     - Caveats of where you think this analysis process might not be capturing the sentiment accurately and suggestions for how to do better in the next round of analysis.
   - What implications your narrative has for the client's goals.
   - High-level explanation of what you did.
2. **Lessons Learned Report:** a report of no more than five pages in Word that includes:
   - For both iPhone and Galaxy:
     - The classifier you selected and the features (attributes) you used to train the classifier.
     - Your rationale for selecting the classifier that you did.
     - Any features you eliminated from the data matrix and your rationale for doing so.
     - Comparative performance of the classifiers you tried (you can explain in text or with a chart).
   - What worked well. What didn't work. What was difficult.What was difficult.
   - How the process to execute similar projects should be changed for the future.

The steps in the following tabs will guide you through this task.

Set Up Parallel Processing

In base configuration, RStudio only uses one core from your computer's processor. This is fine for smaller data sets because computation time is short. The two matrix files you will be using for modeling in this project are relatively large (12,000+ instances x 59 attributes) and performing operations can be time consuming.

In the interest of speeding up processing time, we will set up parallel processing, which means that we will use additional processor cores. Below are the basics for the doParallel package.

```
# Required
library(doParallel)
# Find how many cores are on your machine
detectCores() # Result = Typically 4 to 6
# Create Cluster with desired number of cores. Don't use them all! Your computer is running other processes.
cl <- makeCluster(2)
# Register Cluster
registerDoParallel(cl)
# Confirm how many cores are now "assigned" to R and RStudio
getDoParWorkers() # Result 2
# Stop Cluster. After performing your tasks, stop your cluster.
stopCluster(cl)
```

Get Started – Explore the Data

The workflow of this plan of attack focuses on one small matrix at a time. The plan of attack uses iPhone as the example. Once iPhone modeling and prediction is complete, you should import Galaxy and perform the same steps.

1. Download *iphone_smallmatrix_labeled_8d*.csv and ***galaxy_smallmatrix_labeled_8d*.csv**.

These are the data matrices that you will use to develop your models to predict the overall sentiment toward iPhone and Galaxy. They include

the counts of relevant words (sentiment lexicons) for about 12,000 instances (web pages). The values in the device *sentiment* columns (last column in each matrix) represents the overall sentiment toward the device on a scale of 0-5. The overall sentiment value has been manually input by a team of coworkers who read each webpage and rated the sentiment. The scale is as follows:

0: very negative

1: negative

2: somewhat negative

3: somewhat positive

4: positive

5: very positive

2. Using RStudio, open the ***iphone_smallmatrix_labeled_8d*.csv** file and familiarize yourself with the data. What do the attribute headers and counts represent? Here are some examples to guide you:

1. iOS – counts mentions of iOS on a webpage
2. iphonecampos – counts positive sentiment mentions of the iphone camera
3. galaxydisneg – counts negative sentiment mentions of the Galaxy display
4. htcperunc – counts the unclear sentiment mentions of HTC performance

3. Use the str() and summary() commands. What classes of data are the attributes and y-variable? What is the distribution of the dependent variable? Try plotting it.

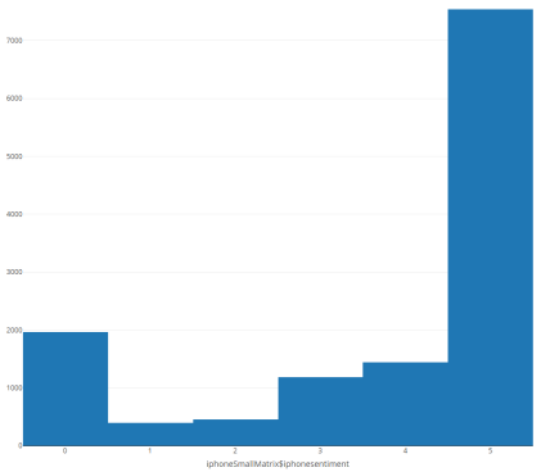
Str()

```
> str(iphoneSmallMatrix)
'data.frame':   12973 obs. of  59 variables:
 $ iphone            : int  1 1 1 1 1 41 1 1 1 1 ...
 $ samsunggalaxy     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ sonyxperia        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nokialumina       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ htcphone          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ ios               : int  0 0 0 0 0 6 0 0 0 0 ...
 $ googleandroid     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonecampos      : int  0 0 0 0 0 1 1 0 0 0 ...
 $ samsungcampos     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ sonycampos        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiacampos       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ htccampos         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonecamneg      : int  0 0 0 0 0 3 1 0 0 0 ...
 $ samsungcamneg     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ sonycamneg        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiacamneg       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ htccamneg         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonecamunc      : int  0 0 0 0 0 7 1 0 0 0 ...
 $ samsungcamunc     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ sonycamunc        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiacamunc       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ htccamunc         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonedispos      : int  0 0 0 0 0 1 13 0 0 0 ...
 $ samsungdispos     : int  0 0 0 0 0 0 0 0 0 0 ...
```

Summary():

```
> summary(iphoneSmallMatrix)
     iphone       samsunggalaxy       sonyxperia        nokialumina          htcphone            ios          
 Min.   :  0.000   Min.   :0.00000   Min.   :0.00000   Min.   :0.000000   Min.   :  0.0000   Min.   :0.0000  
 1st Qu.:  1.000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:  0.0000   1st Qu.:0.0000  
 Median :  1.000   Median :0.00000   Median :0.00000   Median :0.000000   Median :  0.0000   Median :0.0000  
 Mean   :  2.148   Mean   :0.07115   Mean   :0.02405   Mean   :0.002312   Mean   :  0.1371   Mean   :0.1523  
 3rd Qu.:  1.000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:  0.0000   3rd Qu.:0.0000  
 Max.   :46.000   Max.   :8.00000   Max.   :8.00000   Max.   :2.000000   Max.   :479.0000   Max.   :6.0000  
 googleandroid      iphonecampos      samsungcampos       sonycampos        nokiacampos         htccampos       
 Min.   :0.00000   Min.   :  0.0000   Min.   : 0.00000   Min.   :0.000000   Min.   : 0.00000   Min.   :  0.0000  
 1st Qu.:0.00000   1st Qu.:  0.0000   1st Qu.: 0.00000   1st Qu.:0.000000   1st Qu.: 0.00000   1st Qu.:  0.0000  
 Median :0.00000   Median :  0.0000   Median : 0.00000   Median :0.000000   Median : 0.00000   Median :  0.0000  
 Mean   :0.03962   Mean   :  0.2896   Mean   : 0.05373   Mean   :0.009944   Mean   : 0.00686   Mean   :  0.1132  
 3rd Qu.:0.00000   3rd Qu.:  0.0000   3rd Qu.: 0.00000   3rd Qu.:0.000000   3rd Qu.: 0.00000   3rd Qu.:  0.0000  
 Max.   :6.00000   Max.   :156.0000   Max.   :65.00000   Max.   :8.000000   Max.   :17.00000   Max.   :156.0000  
  iphonecamneg      samsungcamneg       sonycamneg        nokiacamneg          htccamneg         iphonecamunc     
 Min.   : 0.0000   Min.   : 0.00000   Min.   :0.000000   Min.   : 0.000000   Min.   : 0.00000   Min.   : 0.0000  
 1st Qu.: 0.0000   1st Qu.: 0.00000   1st Qu.:0.000000   1st Qu.: 0.000000   1st Qu.: 0.00000   1st Qu.: 0.0000  
 Median : 0.0000   Median : 0.00000   Median :0.000000   Median : 0.000000   Median : 0.00000   Median : 0.0000  
 Mean   : 0.2346   Mean   : 0.05473   Mean   :0.002312   Mean   : 0.006167   Mean   : 0.09296   Mean   : 0.2535  
 3rd Qu.: 0.0000   3rd Qu.: 0.00000   3rd Qu.:0.000000   3rd Qu.: 0.000000   3rd Qu.: 0.00000   3rd Qu.: 0.0000  
```

```
plot_ly(df, x= ~df$iphonesentiment, type='histogram')
```



4. Check for missing data and address if necessary.

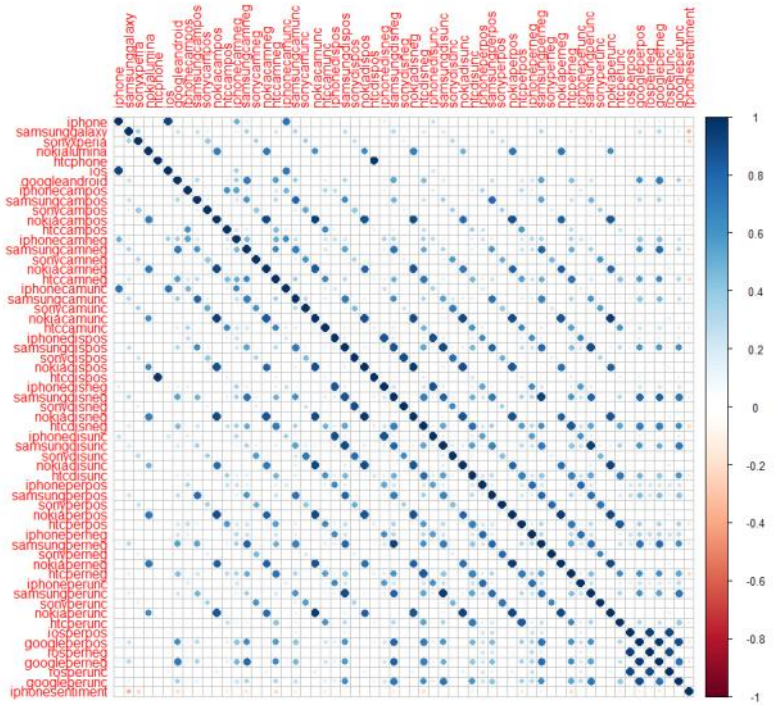
Preprocessing & Feature Selection

Feature Selection

In previous projects you performed feature selection on one data set and then used that data set for modeling. For this project you will create a new data set for each feature selection method. You will then model with each of these new data sets to determine which method, if any, provides the best model accuracy for this project. Let's explore the data set with feature selection in mind and then pre-process.

Examine Correlation (Do Classification problems suffer from Collinearity?)

Use the cor() and corrplot() functions (see C2/T3 if you need a reminder) to understand the correlations with the dependant variable. Note any highly correlated features for removal.




```
#Using the function findcorrelation to see the colleration between the variables
findCorrelation(corrData, cutoff = 0.9,
                verbose = FALSE,
                names = TRUE,
                exact = ncol(corrData) < 100)
#From the analsysis, we are going to delete the independent variables that has a correlation
#up of 90 in order to avoid collinearity
```

```
[1] "samsungdisneg" "samsungdispos" "googleperneg"  "samsungdisunc" "nokiacamunc"   "nokiadisneg"   "nokiaperunc"
[8] "nokiaperneg"   "nokiacamneg"   "iosperunc"     "iosperneg"     "ios"           "htcphone"
```

TIP:
Tip: While printing your cor() output you might see a message about reaching "max print" and a truncated matrix. You can solve this by increasing "max print" with options(). The options() function lets you control a wide variety of global options.

```
options(max.print=1000000)
```

After identifying features for removal, create a new data set if needed. If there are no highly correlated features with the dependant, move on to Feature Variance.

```
# create a new data set and remove features highly correlated with the dependant

iphoneCOR <- iphoneDF

iphoneCOR$featureToRemove <- NULL
```

Examine Feature Variance

The distribution of values within a feature is related to how much information that feature holds in the data set. Features with no variance can be said to hold little to no information. Features that have very little, or "near zero variance", may or may not have useful information. To explore

feature variance we can use nearZeroVar() from the caret package.

```
#nearZeroVar() with saveMetrics = TRUE returns an object containing a table including: frequency ratio, percentage unique, zero variance and near zero variance

nzvMetrics <- nearZeroVar(iphoneDF, saveMetrics = TRUE)

nzvMetrics
```

```
       C:\Users\lmcarva\AppData\Local\Temp\RtmpMfJGpe\downloaded_package
> #nearZeroVar() with saveMetrics = TRUE returns an object containing a t
> #percentage unique, zero variance and near zero variance
> nzvMetrics <- nearZeroVar(iphoneCOR, saveMetrics = TRUE)
> nzvMetrics
                 freqRatio percentUnique zeroVar   nzv
iphone            5.041322    0.20812457   FALSE FALSE
samsunggalaxy    14.127336    0.05395822   FALSE FALSE
sonyxperia       44.170732    0.03854159   FALSE  TRUE
nokialumina     497.884615    0.02312495   FALSE  TRUE
googleandroid    61.247573    0.04624990   FALSE  TRUE
iphonecampos     10.524697    0.23124952   FALSE FALSE
samsungcampos    93.625000    0.08479149   FALSE  TRUE
sonycampos      348.729730    0.05395822   FALSE  TRUE
nokiacampos    1850.142857    0.08479149   FALSE  TRUE
htccampos        79.272152    0.16958298   FALSE  TRUE
iphonecamneg     19.517529    0.13104139   FALSE  TRUE
samsungcamneg   100.132812    0.06937486   FALSE  TRUE
sonycamneg     1851.285714    0.04624990   FALSE  TRUE
htccamneg        93.444444    0.11562476   FALSE  TRUE
iphonecamunc     16.764205    0.16187466   FALSE FALSE
samsungcamunc    74.308140    0.06937486   FALSE  TRUE
```

Review your table. Are there features that have zero variance? Near zero variance? Let's use nearZeroVar() again to create an index of near zero variance features. The index will allow us to quickly remove features.

```
# nearZeroVar() with saveMetrics = FALSE returns an vector

nzv <- nearZeroVar(iphoneDF, saveMetrics = FALSE)

Nzv
```

```
iphonesentiment    3.843017    0.04624990   FALSE FALSE
> # nearZeroVar() with saveMetrics = FALSE returns an vector
> nzv <- nearZeroVar(iphoneCOR, saveMetrics = FALSE)
> nzv
 [1]  3  4  5  7  8  9 10 11 12 13 14 16 17 18 20 21 22 24 25 27 28 29 31 32 33 34 36 37 38 40 41 42 43 44 45
>
```

Does your "nzv" object align with your "nzvMetrics" results?

After identifying features for removal, create a new data set.

```
# create a new data set and remove near zero variance features

iphoneNZV <- iPhoneDF[,-nzv]

str(iphoneNZV)
```

```
> str(iphoneNZV)
'data.frame':   12973 obs. of  11 variables:
 $ iphone         : int  1 1 1 1 1 41 1 1 1 1 ...
 $ samsunggalaxy  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonecampos   : int  0 0 0 0 0 1 1 0 0 0 ...
 $ iphonecamunc   : int  0 0 0 0 0 7 1 0 0 0 ...
 $ iphonedispos   : int  0 0 0 0 0 1 13 0 0 0 ...
 $ iphonedisneg   : int  0 0 0 0 0 3 10 0 0 0 ...
 $ iphonedisunc   : int  0 0 0 0 0 4 9 0 0 0 ...
 $ iphoneperpos   : int  0 1 0 1 1 0 5 3 0 0 ...
 $ iphoneperneg   : int  0 0 0 0 0 0 4 1 0 0 ...
 $ iphoneperunc   : int  0 0 0 1 0 0 5 0 0 0 ...
 $ iphonesentiment: int  0 0 0 0 0 4 4 0 0 0 ...
>
```

Recursive Feature Elimination

RFE is a form of automated feature selection. Caret's rfe() function with random forest will try every combination of feature subsets and return a final list of recommended features.

```
# Let's sample the data before using RFE

set.seed(123)

iphoneSample <- iphoneDF[sample(1:nrow(iphoneDF), 1000, replace=FALSE),]


# Set up rfeControl with randomforest, repeated cross validation and no updates

ctrl <- rfeControl(functions = rfFuncs,

                   method = "repeatedcv",
```

```
                    repeats = 5,
                    verbose = FALSE)
# Use rfe and omit the response variable (attribute 59 iphonesentiment)
rfeResults <- rfe(iphoneSample[,1:58],
                  iphoneSample$iphonesentiment,
                  sizes=(1:58),
                  rfeControl=ctrl)
# Get results
rfeResults
# Plot results
plot(rfeResults, type=c("g", "o"))
```

The resulting table and plot display each subset and its accuracy and kappa. An asterisk denotes the the number of features that is judged the most optimal from RFE.

After identifying features for removal, create a new data set and add the dependant variable.

```
# create new data set with rfe recommended features
iphoneRFE <- iphoneDF[,predictors(rfeResults)]


# add the dependent variable to iphoneRFE
iphoneRFE$iphonesentiment <- iphoneDF$iphonesentiment


# review outcome
str(iPhoneRFE)
```

Other Cases

- **Domain Expertise -** Individual that are highly knowledgeable about the contents of their data may already have a sense of which features are important to modeling. Removing features based on prior research and/or experimentation is common.
- **Unique Identifiers** – These features are often removed because they can't be meaningfully compared.
- **Variable Importance** – You used Caret's varImp() function in C2/T2 to get a ranked list of features from a decision tree model. The ranked list can be used to select features.

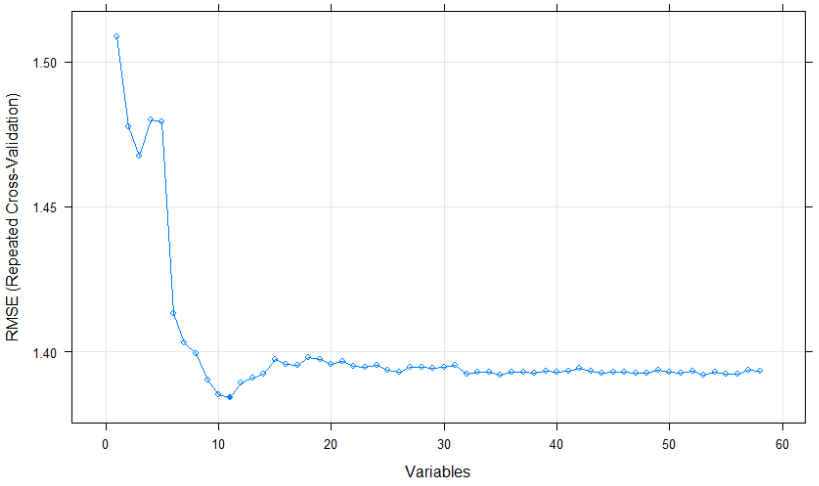
Preprocessing

In Step 2 you used str() to understand the data types in your iPhone small matrix. You also received information that maps the values of the dependent variable. What does the scale from the data map imply? Should iphonesentiment be numeric or something else?

Make any changes to the dependent variable you deem necessary. You need to do this for your original data set and any additional data sets you created in feature selection.

After preprocessing you may have the following data sets:

- iphoneDF (this data set retains all of the original features for "out of the box" modeling)
- iphoneCOR
- iphoneNZV
- iphoneRFE



Model Development and Evaluation

Now that you have performed feature selection, preprocessing and have become familiar with the small matrix file it is time to model. Initially you will model with the data set containing all features to gain "out of the box" accuracy and kappa. Then you will model again with your feature selected data sets.

Your goal is to find the best combination of data set and algorithm as measured by resulting performance metrics

"Out of the Box" Model Development

1. If needed, revisit the Caret Work Flow available in C2/T2.
2. Use your preprocessed small matrix that contains all of the features
3. Set your seed (if you haven't already)
4. A note on sampling: You have previously used sampling as part of your modeling process to increase computational efficiency. In this case, the Alert! Team has painstakingly labeled the iPhone and Galaxy small matrix files. Although you will be sacrificing computational efficiency, you should consider using all observations from these files to build the most complete models possible.
5. Create training and testing sets with a 70/30 split using createDataPartition.
6. Train four algorithms. Model codes for caret can be found here: http://topepo.github.io/caret/available-models.html
   - C5.0

```
> c50modelIphoneRPE
C5.0

9732 samples
  11 predictor
   6 classes: '0', '1', '2', '3', '4', '5'

Pre-processing: centered (11), scaled (11)
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 8759, 8759, 8758, 8759, 8760, 8759, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.7604836  0.5346862
  rules  FALSE   10      0.7564756  0.5321000
  rules  FALSE   20      0.7564756  0.5321000
  rules   TRUE    1      0.7602781  0.5344613
  rules   TRUE   10      0.7556534  0.5290614
  rules   TRUE   20      0.7556534  0.5290614
  tree   FALSE    1      0.7609973  0.5366836
  tree   FALSE   10      0.7529823  0.5270115
  tree   FALSE   20      0.7529823  0.5270115
  tree    TRUE    1      0.7604834  0.5355659
  tree    TRUE   10      0.7533934  0.5280760
  tree    TRUE   20      0.7533934  0.5280760

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were trials = 1, model = tree and winnow = FALSE.
> #Testime
> c50modelTimeIphoneRPE
   user  system elapsed
   1.09    0.05   10.99
```

   - Random Forest

```
> rfmodelIphoneRPE
Random Forest

9732 samples
  11 predictor
   6 classes: '0', '1', '2', '3', '4', '5'

Pre-processing: centered (11), scaled (11)
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 8760, 8760, 8760, 8758, 8759, 8759, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.7444602  0.4878764
   6    0.7618269  0.5433884
  11    0.7538101  0.5321054

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 6.
> #Testtime
> rfmodelTimeIphoneRPE
   user  system elapsed
   7.03    0.14   54.88
```

- SVM (from the e1071 package)

```
> svm_modelIphoneRPE

Call:
svm(formula = iphonesentiment ~ ., data = trainingIphoneRFE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  4625

> #Testtime
> svm_modelTimeIphoneRPE
   user  system elapsed
  14.14    0.08   14.23
> #Need addional step do evaluate the accuracy and the Kappa, using the training data set, an Post Resamp
> svm_modelIphoneRPEModel <- predict(svm_modelIphoneRPE, newdata = trainingIphoneRFE)
> postResampleDataSVNIphoneRPE<-postResample(pred=svm_modelIphoneRPEModel,obs=trainingIphoneRFE$iphonesentiment)
> #This is the value of the model, due to I use training in the Dataset
> postResampleDataSVNIphoneRPE
 Accuracy     Kappa
0.7216779 0.4443559
```

-
- kknn (from the kknn package)

```
> #training results
> knnmodelIphoneRPE

Call:
train.kknn(formula = iphonesentiment ~ ., data = trainingIphoneRFE)

Type of response variable: nominal
Minimal misclassification: 0.6743732
Best kernel: optimal
Best k: 11
> #Testtime
> knnmodelTimeIphoneRPE
   user  system elapsed
   4.19    0.04    4.24
> #Need addional step do evaluate the accuracy and the Kappa, using the training data set, an Post Resamp
> knnmodelIphoneRPEModel <- predict(c50modelIphoneRPE, newdata = trainingIphoneRFE)
> postResampleDataknnIphoneRPE<-postResample(pred=knnmodelIphoneRPEModel,obs=trainingIphoneRFE$iphonesentiment)
> #This is the value of the model, due to I use training in the Dataset
> postResampleDataknnIphoneRPE
 Accuracy     Kappa
0.7674777 0.5504647
```

- Feel free to experiment with other algorithms

1. Which model(s) performed best? Were any of the results very similar? Note which ones.

Algo name	TestTime	Accuracy	Kappa
C5.0	**1.09**	**0.76099**	**0.5366**
RF	7.03	0.7618	0.5433
SVM	14.14	0.7216	0.4443
Knn	4.19	0.7674	0.5504

1. The resutls are similar, so for test time I prefer to chose C5.0

2. Use the predict() function and test set with each of your models

```
322    0   15  226  128 2550
> summary(c50modelIphoneRPEPredition)
   0    1    2    3    4    5
322    0   15  226  128 2550
> summary(rfmodelIphoneRPEPredition)
   0    1    2    3    4    5
335    2   20  232  134 2518
> summary(svm_modelIphoneRPEPredition)
   0    1    2    3    4    5
324    0    0  134  121 2662
> summary(knnmodelIphoneRPEPredition)
   0    1    2    3    4    5
322    0   15  226  128 2550
>
```

1.
3. Learn how well your predictions align with ground truth with postResample().

```
> ################################################################################
> # Comparing PostResamp
> ################################################################################
> #Predition of the model c50
> postResampleDataC50IphoneRPE
 Accuracy     Kappa
0.7528541 0.5214479
> #Predition of the model RF
> postResampleDataRFIphoneRPE
 Accuracy     Kappa
0.7580994 0.5358190
> #Predition of the model SVN
> postResampleDataSVNIphoneRPE
 Accuracy     Kappa
0.7216779 0.4443559
> #Predition of the model Knn
> postResampleDataKknnIphoneRPE
 Accuracy     Kappa
0.7674777 0.5504647
```

Algo name	Accuracy	Kappa
C5.0	0.7528	0.5214
RF	0.7580	0.5358
SVM	0.7216	0.4443
Knn	0.7674	0.5504

In summary the best model that is been chosen is C5.0, due to it represent a similar behavior as the best model that is KNN, but is 4 times faster, and the test time is very important.

Apply Model to Data

Based on comparison of accuracy, kappa and confusion matrix metrics you should now know your best model. In this step you will import your Large Matrix and make predictions with that model.

1.
```
> ################################################################################
> c50modelIphoneRPE
C5.0

9083 samples
  11 predictor
   6 classes: '0', '1', '2', '3', '4', '5'

Pre-processing: centered (11), scaled (11)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 8173, 8176, 8176, 8174, 8176, 8173, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.7226756  0.5005111
  rules  FALSE   10      0.7169857  0.4948869
  rules  FALSE   20      0.7169857  0.4948869
  rules   TRUE    1      0.7237770  0.5018008
  rules   TRUE   10      0.7190778  0.4933880
  rules   TRUE   20      0.7190778  0.4933880
  tree   FALSE    1      0.7195575  0.4998158
  tree   FALSE   10      0.7154815  0.4947196
  tree   FALSE   20      0.7154815  0.4947196
  tree    TRUE    1      0.7197409  0.5000601
  tree    TRUE   10      0.7151513  0.4937692
  tree    TRUE   20      0.7151513  0.4937692

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were trials = 1, model = rules and winnow = TRUE.
```

2.
```
> c50modeliphoneCOR
C5.0

9083 samples
  45 predictor
   6 classes: '0', '1', '2', '3', '4', '5'

Pre-processing: centered (45), scaled (45)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 8173, 8176, 8176, 8174, 8176, 8173, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.7346031  0.5261463
  rules  FALSE   10      0.7416552  0.5185858
  rules  FALSE   20      0.7416552  0.5185858
  rules   TRUE    1      0.7348601  0.5247423
  rules   TRUE   10      0.7255741  0.5079879
  rules   TRUE   20      0.7255741  0.5079879
  tree   FALSE    1      0.7331345  0.5245624
  tree   FALSE   10      0.7418198  0.5235268
  tree   FALSE   20      0.7418198  0.5235268
  tree    TRUE    1      0.7467213  0.5323150
  tree    TRUE   10      0.7226391  0.5088964
  tree    TRUE   20      0.7226391  0.5088964

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were trials = 1, model = tree and winnow = TRUE.
>
```

3.
```
> c50modeliphoneNZV
C5.0

9083 samples
  11 predictor
   6 classes: '0', '1', '2', '3', '4', '5'

Pre-processing: centered (11), scaled (11)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 8173, 8176, 8176, 8174, 8176, 8173, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.7580094  0.5246103
  rules  FALSE   10      0.7442477  0.5010234
  rules  FALSE   20      0.7442477  0.5010234
  rules   TRUE    1      0.7571654  0.5230986
  rules   TRUE   10      0.7439546  0.4992762
  rules   TRUE   20      0.7439546  0.4992762
  tree   FALSE    1      0.7575323  0.5244405
  tree   FALSE   10      0.7437324  0.5011821
  tree   FALSE   20      0.7437324  0.5011821
  tree    TRUE    1      0.7569081  0.5233489
  tree    TRUE   10      0.7416756  0.4972043
  tree    TRUE   20      0.7416756  0.4972043

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were trials = 1, model = rules and winnow = FALSE.
>
```

4.
```
> c50modeliphoneDF
C5.0

9083 samples
  58 predictor
   6 classes: '0', '1', '2', '3', '4', '5'

Pre-processing: centered (58), scaled (58)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 8173, 8175, 8174, 8173, 8177, 8174, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.6839953  0.4776578
  rules  FALSE   10      0.7274632  0.5097309
  rules  FALSE   20      0.7274632  0.5097309
  rules   TRUE    1      0.7136993  0.5020277
  rules   TRUE   10      0.7208765  0.5027922
  rules   TRUE   20      0.7208765  0.5027922
  tree   FALSE    1      0.6898199  0.4814912
  tree   FALSE   10      0.7414930  0.5225840
  tree   FALSE   20      0.7414930  0.5225840
  tree    TRUE    1      0.7068085  0.4942078
  tree    TRUE   10      0.7241797  0.5072864
  tree    TRUE   20      0.7241797  0.5072864

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were trials = 10, model = tree and winnow = FALSE.
>
```

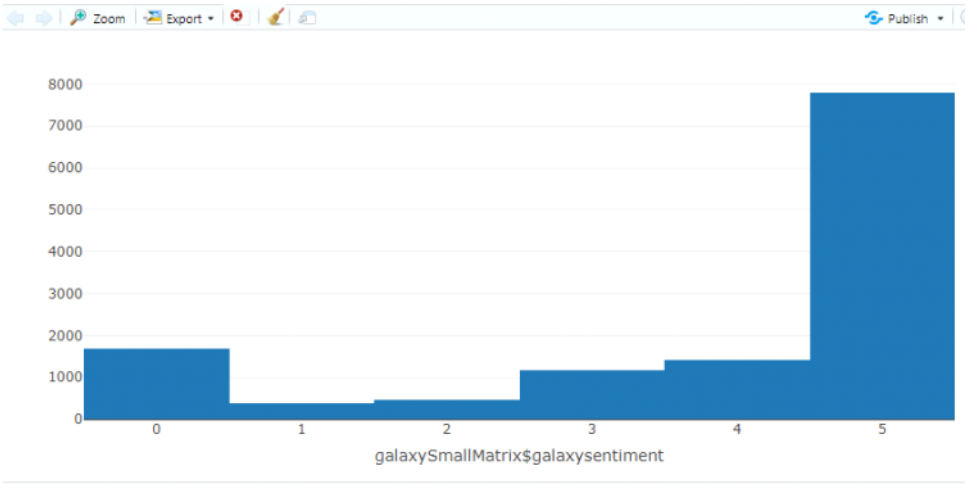
5. Data

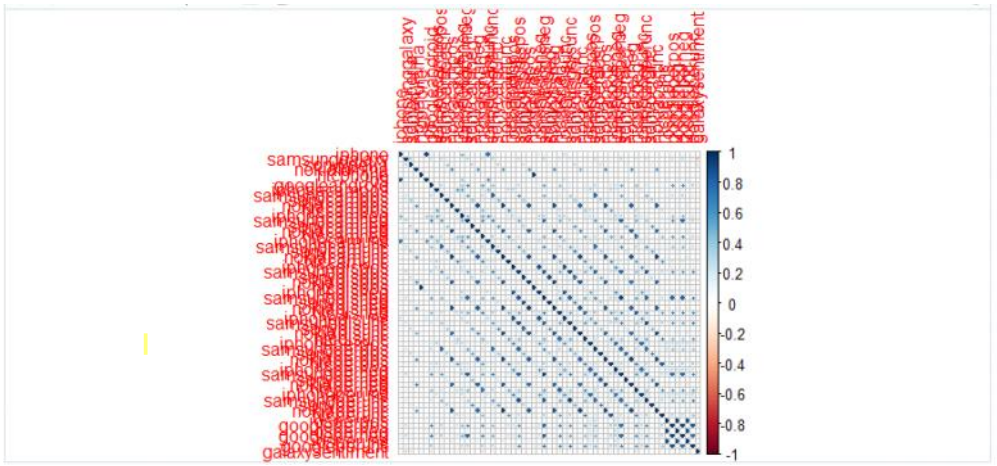
DataSetame	Testtime	Accuracy	Kappa
Recursive Feature Elimination	1.27	0.7237	0.5018
Correlation Elimination	2.56	0.7467	0.5323
Feature Variance(NZV)	**1.36**	**0.7580**	**0.5246**
RC	7.22	0.7997	0.5630

6. Open your LargeMatrix with excel and create a column header on the far right called iphonesentiment. This must exactly match the header from the iphone small matrix. Save the file as iphoneLargeMatrix.
7. Import the iphoneLargeMatrix into RStudio
8. Any feature selection you have done to the iphone small matrix that created your best model must also be done to the Large Matrix.
9. Use predict(), your best model and iphoneLargeMatrix to predict sentiment
10. The summary() function and your prediction object will give you the sentiment counts for each sentiment level

Now it's time focus on the galaxy small matrix. This is a separate data set. The best performing model with the iphone small matrix may not be the best one for galax

Now moving to galaxy





Choosing model,

```
> #Printing the model already trained
> c50modelgalaxyRPE
C5.0

9040 samples
  58 predictor
   6 classes: '0', '1', '2', '3', '4', '5'

Pre-processing: centered (58), scaled (58)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 8137, 8135, 8136, 8136, 8137, 8134, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.6229562  0.4075114
  rules  FALSE   10      0.6972902  0.4633277
  rules  FALSE   20      0.6972902  0.4633277
  rules   TRUE    1      0.6327253  0.4126289
  rules   TRUE   10      0.6493001  0.4178424
  rules   TRUE   20      0.6493001  0.4178424
  tree   FALSE    1      0.6170585  0.4036689
  tree   FALSE   10      0.7128034  0.4758073
  tree   FALSE   20      0.7128034  0.4758073
  tree    TRUE    1      0.6157676  0.4019204
  tree    TRUE   10      0.6710565  0.4393842
  tree    TRUE   20      0.6710565  0.4393842

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were trials = 10, model = tree and winnow = FALSE.
> #Testime
> c50modelTimegalaxyRPE
   user  system elapsed
   5.70    0.09  134.89
```

```
> rfmodelgalaxyRPE
Random Forest

9686 samples
  58 predictor
   6 classes: '0', '1', '2', '3', '4', '5'

Pre-processing: centered (58), scaled (58)
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 8717, 8717, 8718, 8716, 8719, 8719, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.7037981  0.3534424
  30    0.7640905  0.5302536
  58    0.7569652  0.5192793

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 30.
> #Testtime
> rfmodelTimegalaxyRPE
   user  system elapsed
  77.99    0.23  755.21


> svm_modelgalaxyRPE

Call:
svm(formula = galaxysentiment ~ ., data = traininggalaxyRFE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  4866

> #Testtime
> svm_modelTimegalaxyRPE
   user  system elapsed
  55.78    0.30   56.37
> #Need addional step do evaluate the accuracy and the Kappa, using the training data set, an Post Resamp
> svm_modelgalaxyRPEModel <- predict(svm_modelgalaxyRPE, newdata = traininggalaxyRFE)
> postResampleDataSVNgalaxyRPE<-postResample(pred=svm_modelgalaxyRPEModel,obs=traininggalaxyRFE$galaxysentiment)
> #This is the value of the model, due to I use training in the Dataset
> postResampleDataSVNgalaxyRPE
 Accuracy     Kappa
0.7130531 0.3923195
```

```
> knnmodelgalaxyRPE

Call:
train.kknn(formula = galaxysentiment ~ ., data = traininggalaxyRFE)

Type of response variable: nominal
Minimal misclassification: 0.2467479
Best kernel: optimal
Best k: 11
> #training results
> knnmodelgalaxyRPE

Call:
train.kknn(formula = galaxysentiment ~ ., data = traininggalaxyRFE)

Type of response variable: nominal
Minimal misclassification: 0.2467479
Best kernel: optimal
Best k: 11
> #Testtime
> knnmodelTimegalaxyRPE
   user  system elapsed
  13.09    0.04   13.19
> #Need addional step do evaluate the accuracy and the Kappa, using the training data set, an Post Resamp
> knnmodelgalaxyRPEModel <- predict(c50modelgalaxyRPE, newdata = traininggalaxyRFE)
> postResampleDatakknngalaxyRPE<-postResample(pred=knnmodelgalaxyRPEModel,obs=traininggalaxyRFE$galaxysentiment)
> #This is the value of the model, due to I use training in the Dataset
> postResampleDatakknngalaxyRPE
 Accuracy     Kappa
0.7642699 0.5298924
```

Algo name	TestTime	Accuracy	Kappa
C5.0	5.70	0.7128	0.4758
RF	77.99	0.76409	0.5302
SVM	55.78	0.7130	0.3923
Knn	13.09	0.7642	0.5298

```
> ##############################################################################
> # Comparing PostResamp
> ##############################################################################
> #C50 Model PostResamp
> postResampleDataC50galaxyRPE
 Accuracy     Kappa
0.7702326 0.5416845
> #RF Model PostResamp
> postResampleDataRFgalaxyRPE
 Accuracy     Kappa
0.7742636 0.5534864
> #SVN Model PostResamp
> postResampleDataSVNgalaxyRPE
 Accuracy     Kappa
0.7130531 0.3923195
> #KNN Model PostResamp
> postResampleDataknngalaxyRPE
 Accuracy     Kappa
0.7642699 0.5298924
```

Algo name	Accuracy	Kappa
C5.0	0.77023	0.5416
RF	0.7742	0.5534
SVM	0.7130	0.3923
Knn	0.7642	0.5298

Now running with C5 algo for RPE Data Set

```
> ##############################################################################
> #Comparing the resutls
> ##############################################################################
> c50modelgalaxyRPE
C5.0

9040 samples
  58 predictor
   6 classes: '0', '1', '2', '3', '4', '5'

Pre-processing: centered (58), scaled (58)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 8137, 8135, 8136, 8136, 8137, 8134, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.6229562  0.4075114
  rules  FALSE   10      0.6972902  0.4633277
  rules  FALSE   20      0.6972902  0.4633277
  rules   TRUE    1      0.6327253  0.4126289
  rules   TRUE   10      0.6493001  0.4178424
  rules   TRUE   20      0.6493001  0.4178424
  tree   FALSE    1      0.6170585  0.4036689
  tree   FALSE   10      0.7128034  0.4758073
  tree   FALSE   20      0.7128034  0.4758073
  tree    TRUE    1      0.6157676  0.4019204
  tree    TRUE   10      0.6710565  0.4393842
  tree    TRUE   20      0.6710565  0.4393842

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were trials = 10, model = tree and winnow = FALSE.
```

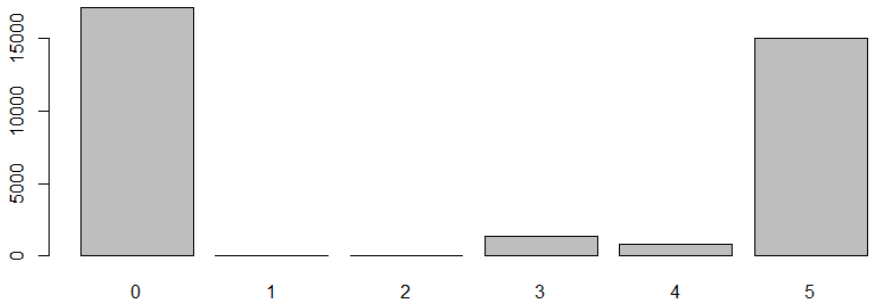
Now running with C5 algo for COR Data Set

```
> c50modelgalaxyCOR
C5.0

9040 samples
  45 predictor
   6 classes: '0', '1', '2', '3', '4', '5'

Pre-processing: centered (45), scaled (45)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 8137, 8135, 8136, 8136, 8137, 8134, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.6225147  0.4072230
  rules  FALSE   10      0.7135321  0.4738299
  rules  FALSE   20      0.7135321  0.4738299
  rules   TRUE    1      0.6362256  0.4130025
  rules   TRUE   10      0.6911719  0.4531930
  rules   TRUE   20      0.6911719  0.4531930
  tree   FALSE    1      0.6151774  0.4012022
  tree   FALSE   10      0.6948255  0.4637648
  tree   FALSE   20      0.6948255  0.4637648
  tree    TRUE    1      0.6150305  0.4011905
  tree    TRUE   10      0.6725539  0.4420170
  tree    TRUE   20      0.6725539  0.4420170

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were trials = 10, model = rules and winnow = FALSE.
> e
```

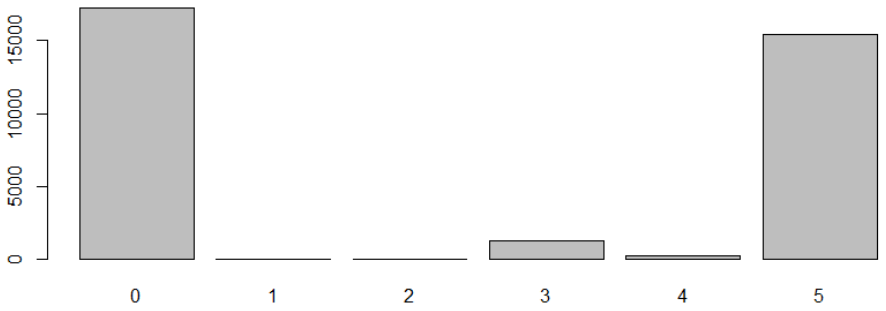
```
> c50modelgalaxyNZV #Best Model
C5.0

9040 samples
  11 predictor
   6 classes: '0', '1', '2', '3', '4', '5'

Pre-processing: centered (11), scaled (11)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 8137, 8135, 8136, 8136, 8137, 8134, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.7482267  0.4883466
  rules  FALSE   10      0.7379383  0.4624429
  rules  FALSE   20      0.7379383  0.4624429
  rules   TRUE    1      0.7476738  0.4875969
  rules   TRUE   10      0.7358390  0.4563583
  rules   TRUE   20      0.7358390  0.4563583
  tree   FALSE    1      0.7476001  0.4880146
  tree   FALSE   10      0.7353940  0.4594168
  tree   FALSE   20      0.7353940  0.4594168
  tree    TRUE    1      0.7471948  0.4873465
  tree    TRUE   10      0.7345859  0.4567890
  tree    TRUE   20      0.7345859  0.4567890

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were trials = 1, model = rules and winnow = FALSE.

> c50modelgalaxyDF
C5.0

9040 samples
  58 predictor
   6 classes: '0', '1', '2', '3', '4', '5'

Pre-processing: centered (58), scaled (58)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 8137, 8135, 8136, 8136, 8137, 8134, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.6229194  0.4075068
  rules  FALSE   10      0.6973271  0.4634383
  rules  FALSE   20      0.6973271  0.4634383
  rules   TRUE    1      0.6322062  0.4120540
  rules   TRUE   10      0.6676085  0.4332975
  rules   TRUE   20      0.6676085  0.4332975
  tree   FALSE    1      0.6169848  0.4035710
  tree   FALSE   10      0.7128034  0.4758073
  tree   FALSE   20      0.7128034  0.4758073
  tree    TRUE    1      0.6161363  0.4023690
  tree    TRUE   10      0.6547442  0.4283821
  tree    TRUE   20      0.6547442  0.4283821

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were trials = 10, model = tree and winnow = FALSE.
> # create a new dataset that will be used for recoding sentiment
```

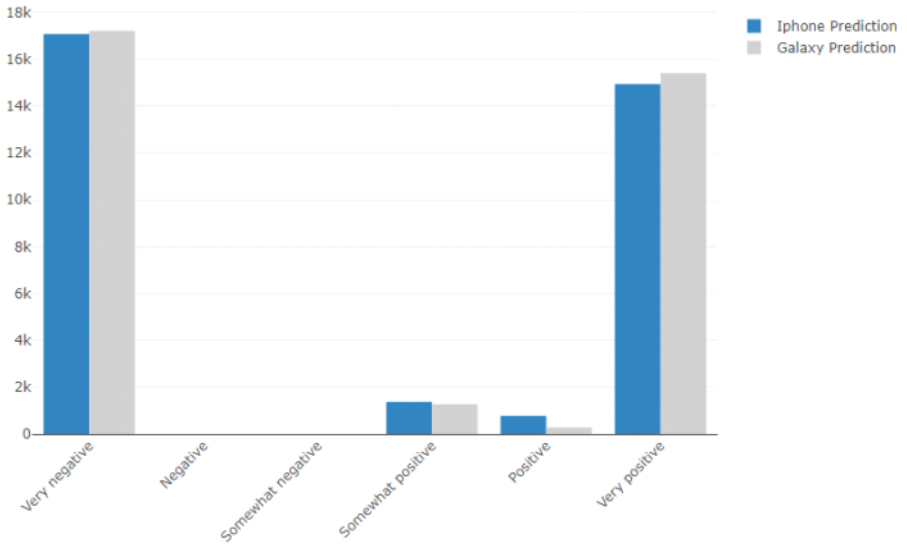
DataSetame	Testtime	Accuracy	Kappa
Recursive Feature Elimination	5.70	0.7128	0.4758
Correlation Elimination	3.54	0.7135	0.4738
Feature Variance(NZV)	1.17	0.7482	0.4883
RC	**3.47**	**0.7808**	**0.5386**

Iphone



Galaxy


0
5000
10000
15000
0
1
2
3
4
5


18k
16k
14k
12k
10k
8k
6k
4k
2k
0
Very negative
Negative
Somewhat negative
Somewhat positive
Positive
Very positive
Iphone Prediction
Galaxy Prediction