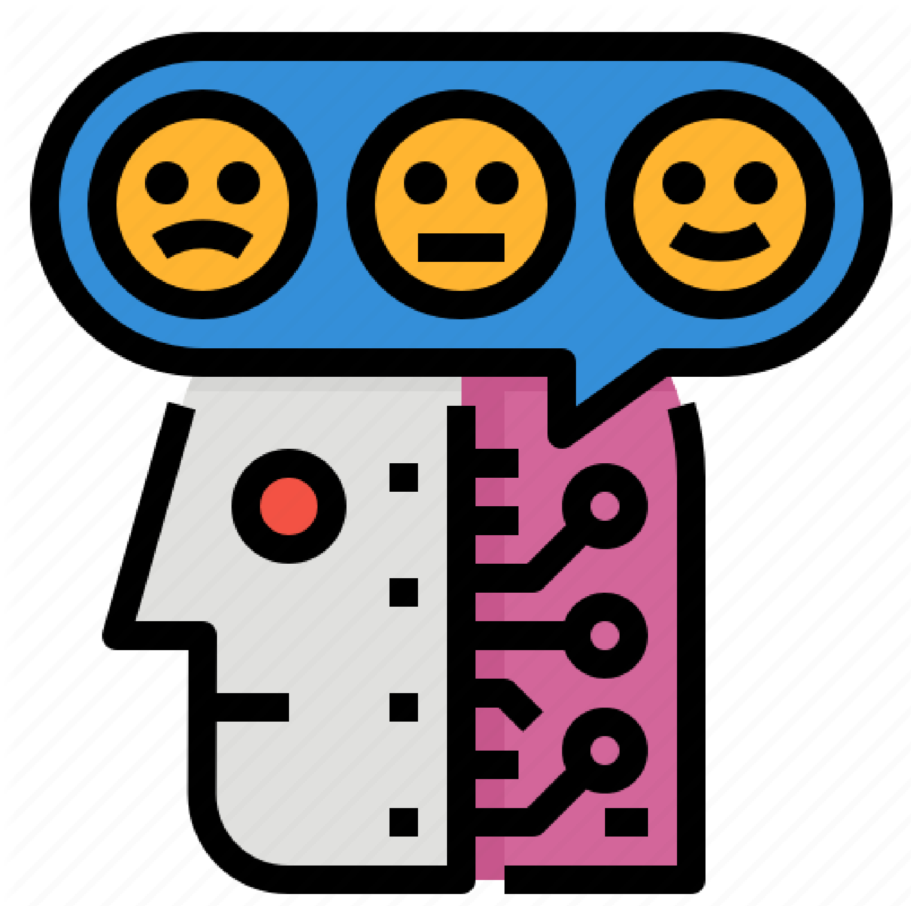
SENTIMENT ANALYSIS OF THE IPHONE AND THE GALAXY



Name of Project: Helio **Client**: Apple and Samsung **Date:** 11/14/2019

SUMMARY

I. OVERVIEW

Being contacted by the Helio project Manager to run a sentiment analysis of the iPhone and the Galaxy smartphones for their clients Apple and Samsung.

While I worked on collecting the Large Matrix using EMR to compile web pages from the Common Crawl that are relevant to smart phones, the Alert! Analytics team did manually label each instance of two small matrices with sentiment toward iPhone and Samsung Galaxy. They read through each webpage and assigned a sentiment rating based on their findings.

The analytic goal for this project was to build models that understand the patterns in the two small matrices and then use those models with the Large Matrix to predict sentiment for iPhone and Galaxy.

In order to accomplish my analysis, I have followed the following steps:

- Setting up parallel processing
- Exploring the Small Matrices to understand the attributes
- Preprocessing & Feature Selection
- Model Development and Evaluation
- Feature Engineering
- Applying Model to Large Matrix and getting Predictions
- Analyzing results, writing up findings report
- Writing lessons learned report

I have used R statistical programming language and the caret package to perform this work. To get the best results, I compared the performance metrics of four different classifiers, namely C5.0, random forest, KKNN and support vector machines. The modeling has been done for both the iPhone and Galaxy data sets.

After comparing the performance of the classifiers in "out of the box modeling, I have done some feature selection/feature engineering in order to improve the performance metrics of the models.

After identifying my most optimal model, I have used it to predict sentiment in the Large Matrix collected as described above.

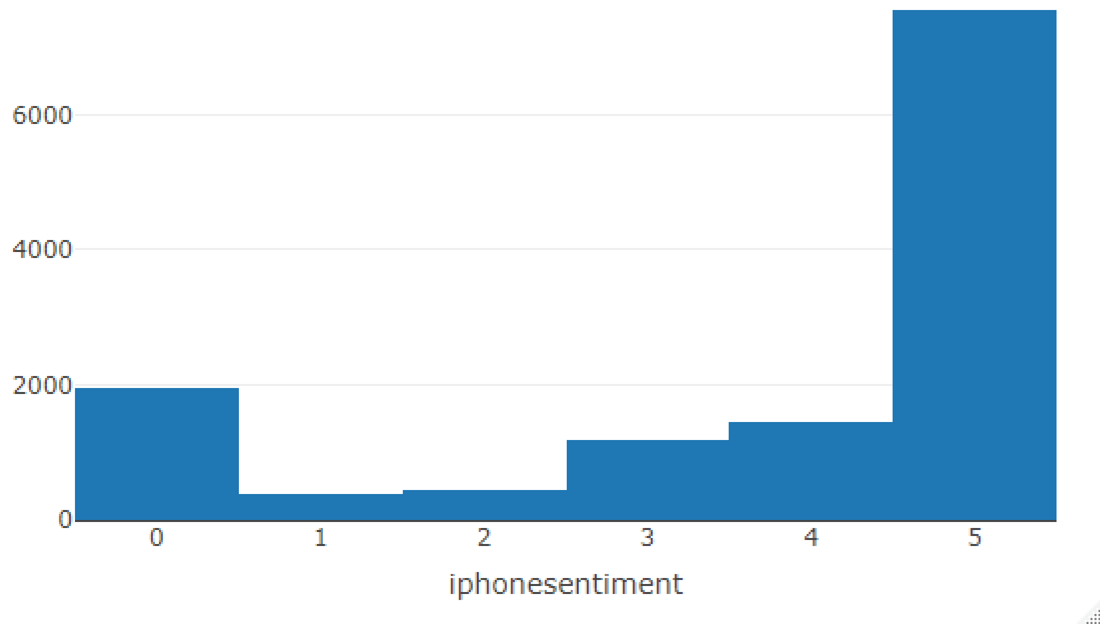
This report summarizes my findings. With an interpretation of the results and the coding.

In addition to the Summary of Findings for Helio, I have prepared a brief Lessons Learned Report presented in the Appendix of this report.

II. IPHONE SMALL MATRIX SENTIMENT ANALYSIS

Distribution Of Dependent Variable

```
> plot_ly(iphone, x= ~iphonesentiment, type='histogram')
```



==

First plotting of the distribution of the dependent variable showed over 6000 of very positive sentiment on the iPhone.

==

Missing Values

```
#No Missing Values Found
> sum(is.na(iphone))
[1] 0
```

Correlation Matrix

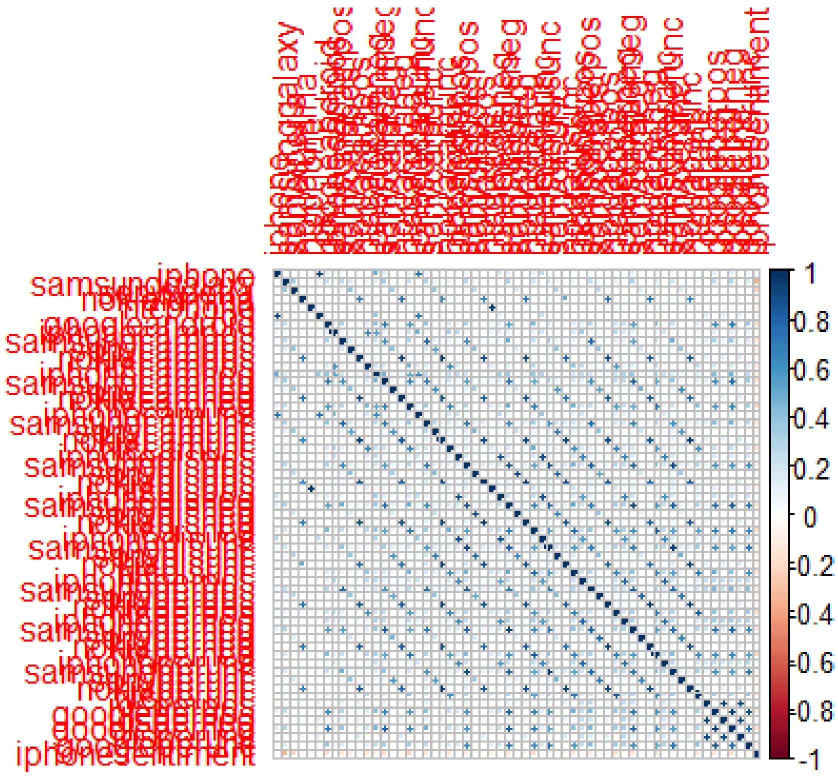
Examining the correlation in the dataset helps determine if they were a relationship between the variables, it also indicates both strength of the relationship as well as the direction positive vs negative or neutral.

The reason why we used this process is because the performance of some algorithms can be deteriorated if two or more variables are tightly related, called multicollinearity. In that case we need to remove one of the offending correlated variables in order to improve the skill of the model.

```
> iphoneCor <- cor(iphoneDF)
                  googleperneg     iosperunc  googleperunc iphonesentiment
iphone            0.1387423597 -0.0203681182  0.0678592347     0.014858654
samsunggalaxy     0.2909745336 -0.0153292775  0.1422517633    -0.359172760
sonyxperia       -0.0085698468 -0.0148024105 -0.0079160304    -0.233169880
nokialumina       0.0006534676  0.0528866876  0.0079987614    -0.055961769
htcphone          0.0207261368 -0.0026662299  0.0133049541    -0.051284868
ios              -0.0180284915  0.1170349668 -0.0102331140     0.001656417
googleandroid     0.7165147742 -0.0163771745  0.3719984702    -0.189142050
iphonecampos      0.1243547611 -0.0010370936  0.0730039141    -0.029731217
samsungcampos     0.3573624065  0.0448897145  0.1591714963    -0.112743311
sonycampos        0.0084545596 -0.0064206991 -0.0034336603    -0.090665090
nokiacampos       0.0039406908  0.1651880784  0.0125178481    -0.033374561
htccampos         0.1772296141 -0.0060785851  0.1000310846    -0.120434115
iphonecamneg      0.4680746932 -0.0107487208  0.2410025125    -0.083963139
samsungcamneg     0.7942819521  0.0470453605  0.3421196381    -0.185988857
sonycamneg        0.0251264773 -0.0028646448 -0.0015319542    -0.024826403
nokiacamneg      -0.0004450102  0.1436761784  0.0037721473    -0.033069469
htccamneg         0.6526440067 -0.0101911803  0.3337273887    -0.222972178
iphonecamunc      0.0748583656 -0.0013364593  0.0581386691     0.001443485
samsungcamunc     0.4766899704  0.0576123258  0.2694315123    -0.138045912
sonycamunc       -0.0042044682 -0.0044625643 -0.0023864893    -0.050326854
nokiacamunc       0.0012988106  0.1735035602  0.0068285408    -0.031549730
htccamunc         0.2275768031 -0.0051864137  0.1624307557    -0.148881468
iphonedispos      0.1470225674  0.0247667616  0.1796863033     0.014546824
samsungdispos     0.5799514805  0.0572875968  0.6361031491    -0.099262059
sonydispos        0.0017093490 -0.0035623113 -0.0019050522    -0.038635303
nokiadispos      -0.0026677068  0.1272636476 -0.0015142114    -0.025922378
htcdispos         0.1092388830  0.0005786204  0.1240182192    -0.060405793
iphonedisneg      0.2136404410  0.0182216460  0.2044162638     0.003144905
samsungdisneg     0.8266039307  0.0507296081  0.7355261466    -0.139964721
sonydisneg        0.0019065213 -0.0015012009 -0.0008028119    -0.019956110
nokiadisneg      -0.0028569223  0.1394095914 -0.0016216116    -0.028758588
htcdisneg         0.6983625307  0.0098165195  0.6431743675    -0.192727267
iphonedisunc      0.1060835760  0.0301070415  0.1722760448     0.027172723
samsungdisunc     0.5127319014  0.0399508627  0.7384565052    -0.059548267
sonydisunc       -0.0030380429 -0.0032245366 -0.0017244171    -0.032137154
nokiadisunc      -0.0022863969  0.1626898312 -0.0012977769    -0.023971988
htcdisunc         0.4065686064  0.0196765106  0.5934935043    -0.132952797
iphoneperpos      0.2188475265  0.2118092692  0.2376254310     0.029637900
samsungperpos     0.4412290699  0.1370566656  0.4275422572    -0.081063185
sonyperpos        0.0080695911 -0.0044727090 -0.0023919145    -0.038912744
nokiaperpos      -0.0008238620  0.1359424951  0.0031408747    -0.041594613
htcperpos         0.3584582984 -0.0001889917  0.3683266512    -0.178427038
iphoneperneg      0.3486853181  0.2557356982  0.2962268490    -0.004804058
samsungperneg     0.7963651128  0.1031577743  0.6412286948    -0.138656977
sonyperneg        0.0121404799 -0.0022095991 -0.0011816490    -0.030850090
nokiaperneg      -0.0016059114  0.1280016003  0.0016351213    -0.044219386
htcperneg         0.6389410325  0.0003615931  0.5399024601    -0.209196046
iphoneperunc      0.1962538259  0.1817828791  0.2971400100     0.037199859
samsungperunc     0.5415039792  0.0538973132  0.7398874595    -0.057919616
sonyperunc       -0.0026432711 -0.0028055314 -0.0015003415    -0.018084032
nokiaperunc      -0.0001365632  0.1577138419  0.0041802716    -0.036166807
htcperunc         0.2808931406  0.0084370013  0.3945515791    -0.114171252
iosperpos        -0.0106756484  0.9050794409 -0.0060595823    -0.015757978
googleperpos      0.9574098116 -0.0095235760  0.8870329991    -0.137261491
```

```
iosperneg       -0.0101154769  0.8998188712 -0.0057416245    -0.010179313
googleperneg     1.0000000000 -0.0104682305  0.7561184782    -0.163919084
iosperunc       -0.0104682305  1.0000000000 -0.0059418502    -0.011787428
googleperunc     0.7561184782 -0.0059418502  1.0000000000    -0.070284159
iphonesentiment -0.1639190837 -0.0117874280 -0.0702841593     1.000000000
```

```
> Corrplot(iphoneCor)
```



==

From the corrplot figure above, we can see that there are no such highly correlated features with the dependant variable to remove.

==

NearZeroVariance

To explore feature variance we have used **nearZeroVar()** function from the caret package:

```
> nzvMetrics <- nearZeroVar(iphoneDF, 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
htcphone          11.439614    0.06937486   FALSE FALSE
ios               27.735294    0.04624990   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
nokiacamneg     2158.833333    0.06166654   FALSE  TRUE
htccamneg         93.444444    0.11562476   FALSE  TRUE
iphonecamunc      16.764205    0.16187466   FALSE FALSE
samsungcamunc     74.308140    0.06937486   FALSE  TRUE
sonycamunc       588.318182    0.03854159   FALSE  TRUE
nokiacamunc     2591.200000    0.05395822   FALSE  TRUE
htccamunc         50.548000    0.12333308   FALSE  TRUE
iphonedispos       6.792440    0.24666615   FALSE FALSE
samsungdispos     97.061069    0.13104139   FALSE  TRUE
sonydispos       331.076923    0.06937486   FALSE  TRUE
nokiadispos     1438.777778    0.09249981   FALSE  TRUE
htcdispos         64.694301    0.20041625   FALSE  TRUE
iphonedisneg      10.084428    0.18499961   FALSE FALSE
samsungdisneg     99.155039    0.10791644   FALSE  TRUE
sonydisneg      2159.333333    0.06937486   FALSE  TRUE
nokiadisneg     1850.142857    0.08479149   FALSE  TRUE
htcdisneg         88.492958    0.14645803   FALSE  TRUE
iphonedisunc      11.471875    0.20812457   FALSE FALSE
samsungdisunc     74.255814    0.09249981   FALSE  TRUE
sonydisunc       719.222222    0.05395822   FALSE  TRUE
nokiadisunc     1619.375000    0.04624990   FALSE  TRUE
htcdisunc         50.590361    0.13874971   FALSE  TRUE
iphoneperpos       9.297834    0.19270793   FALSE FALSE
samsungperpos     94.200000    0.10791644   FALSE  TRUE
sonyperpos       416.870968    0.06166654   FALSE  TRUE
nokiaperpos     2158.000000    0.08479149   FALSE  TRUE
htcperpos         74.279762    0.19270793   FALSE  TRUE
iphoneperneg      11.054137    0.16958298   FALSE FALSE
samsungperneg    101.650794    0.10020812   FALSE  TRUE
sonyperneg      2159.666667    0.07708317   FALSE  TRUE
nokiaperneg     3237.250000    0.09249981   FALSE  TRUE
htcperneg         94.428571    0.15416635   FALSE  TRUE
iphoneperunc      13.018349    0.12333308   FALSE FALSE
samsungperunc     86.500000    0.09249981   FALSE  TRUE
sonyperunc      3240.250000    0.04624990   FALSE  TRUE
nokiaperunc     1850.428571    0.06937486   FALSE  TRUE
htcperunc         50.055556    0.15416635   FALSE  TRUE
iosperpos        153.373494    0.09249981   FALSE  TRUE
googleperpos      98.592308    0.06937486   FALSE  TRUE
iosperneg        141.744444    0.09249981   FALSE  TRUE
googleperneg      99.403101    0.08479149   FALSE  TRUE
iosperunc        135.893617    0.07708317   FALSE  TRUE
googleperunc      96.443609    0.07708317   FALSE  TRUE
iphonesentiment    3.843017    0.04624990   FALSE FALSE
```

Recursive Feature Elimination

Caret's **rfe()** function with random forest will try every combination of feature subsets and retur n a final list of recommended features:

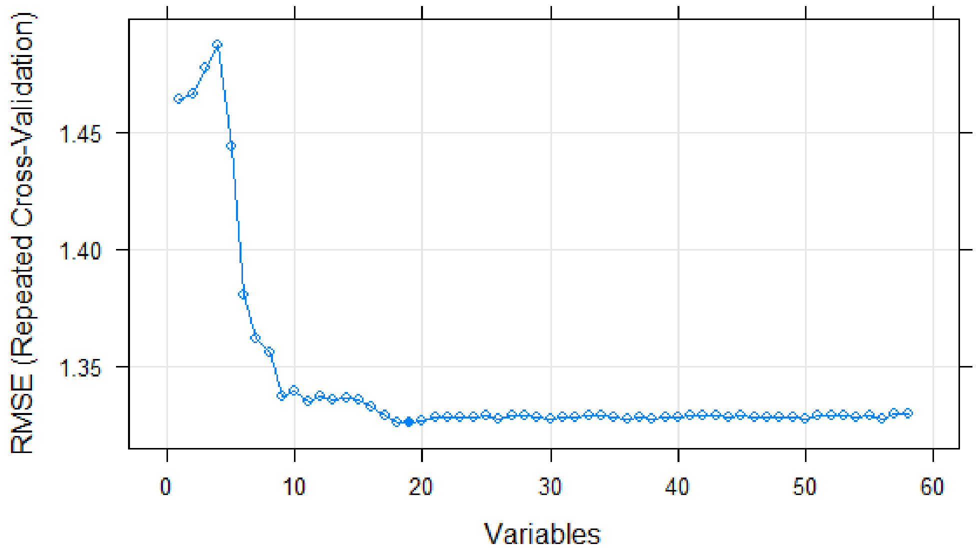
```
> rfeResults <- rfe(iphoneSample[,1:58], iphoneSample$iphonesentiment, sizes=
(1:58), rfeControl=ctrl)
> rfeResults

Recursive feature selection

Outer resampling method: Cross-Validated (10 fold, repeated 5 times)
```

Resampling performance over subset size:

Variables	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD	Selected
1	1.464	0.3158	1.1027	0.1219	0.07907	0.07395	
2	1.466	0.3244	1.1383	0.1189	0.07786	0.07280	
3	1.478	0.3218	1.1565	0.1149	0.07599	0.06821	
4	1.487	0.3208	1.1659	0.1158	0.07742	0.07124	
5	1.444	0.3642	1.1345	0.1128	0.08854	0.06491	
6	1.380	0.3927	1.0184	0.1284	0.09304	0.07729	
7	1.362	0.4081	0.9993	0.1205	0.08973	0.06427	
8	1.356	0.4139	0.9954	0.1211	0.09095	0.06536	
9	1.338	0.4277	0.9438	0.1286	0.09410	0.07156	
10	1.339	0.4268	0.9460	0.1276	0.09211	0.07046	
11	1.335	0.4302	0.9456	0.1240	0.08998	0.06690	
12	1.337	0.4287	0.9242	0.1297	0.09372	0.07402	
13	1.336	0.4298	0.9275	0.1296	0.09418	0.07432	
14	1.336	0.4295	0.9313	0.1286	0.09340	0.07433	
15	1.336	0.4303	0.9170	0.1301	0.09461	0.07819	
16	1.333	0.4321	0.9190	0.1306	0.09559	0.07900	
17	1.329	0.4355	0.9213	0.1294	0.09447	0.07730	
18	1.326	0.4380	0.9095	0.1302	0.09472	0.07795	
19	1.326	0.4379	0.9138	0.1309	0.09539	0.07831	*
20	1.327	0.4375	0.9176	0.1299	0.09468	0.07762	
21	1.328	0.4366	0.9112	0.1314	0.09539	0.07890	
22	1.328	0.4362	0.9147	0.1303	0.09559	0.07768	
23	1.329	0.4360	0.9183	0.1298	0.09500	0.07734	
24	1.328	0.4365	0.9110	0.1304	0.09581	0.07829	
25	1.329	0.4359	0.9145	0.1302	0.09532	0.07843	
26	1.328	0.4365	0.9163	0.1300	0.09533	0.07771	
27	1.329	0.4355	0.9113	0.1295	0.09494	0.07789	
28	1.329	0.4357	0.9134	0.1290	0.09462	0.07760	
29	1.328	0.4363	0.9149	0.1289	0.09492	0.07752	
30	1.328	0.4368	0.9097	0.1295	0.09466	0.07767	
31	1.328	0.4365	0.9125	0.1292	0.09466	0.07660	
32	1.328	0.4366	0.9141	0.1297	0.09541	0.07712	
33	1.329	0.4356	0.9106	0.1294	0.09451	0.07829	
34	1.329	0.4356	0.9126	0.1288	0.09448	0.07739	
35	1.328	0.4366	0.9140	0.1293	0.09481	0.07706	
36	1.328	0.4368	0.9098	0.1300	0.09529	0.07788	
37	1.328	0.4364	0.9115	0.1301	0.09532	0.07762	
38	1.327	0.4372	0.9130	0.1293	0.09501	0.07660	
39	1.328	0.4364	0.9096	0.1304	0.09554	0.07852	
40	1.328	0.4365	0.9114	0.1295	0.09475	0.07754	
41	1.329	0.4358	0.9133	0.1288	0.09422	0.07644	
42	1.329	0.4354	0.9102	0.1301	0.09581	0.07785	
43	1.329	0.4360	0.9108	0.1299	0.09476	0.07755	
44	1.329	0.4360	0.9137	0.1298	0.09532	0.07705	
45	1.329	0.4359	0.9101	0.1299	0.09498	0.07722	
46	1.329	0.4361	0.9110	0.1298	0.09504	0.07733	
47	1.328	0.4364	0.9131	0.1297	0.09492	0.07714	
48	1.329	0.4361	0.9099	0.1309	0.09577	0.07819	
49	1.328	0.4363	0.9112	0.1298	0.09497	0.07748	
50	1.328	0.4366	0.9122	0.1297	0.09470	0.07715	
51	1.329	0.4359	0.9106	0.1306	0.09560	0.07839	
52	1.329	0.4359	0.9110	0.1306	0.09536	0.07798	
53	1.329	0.4358	0.9129	0.1298	0.09487	0.07672	
54	1.329	0.4363	0.9101	0.1297	0.09523	0.07687	
55	1.329	0.4357	0.9109	0.1297	0.09499	0.07737	
56	1.328	0.4366	0.9108	0.1296	0.09461	0.07721	
57	1.330	0.4350	0.9102	0.1290	0.09432	0.07701	
58	1.330	0.4348	0.9111	0.1298	0.09473	0.07762	

The top 5 variables (out of 19):

```
  iphone, googleandroid, iphonedispos, iphonedisneg, samsunggalaxy
> plot(rfeResults, type=c("g", "o"))
```




```
> iphoneRFE <- iphoneDF[,predictors(rfeResults)]
```

Add the independent variable:

```
> iphoneRFE$iphonesentiment <- iphoneDF$iphonesentiment
>
> str(iphoneRFE)
Classes ‘spec_tbl_df’, ‘tbl_df’, ‘tbl’ and 'data.frame':     12973 obs. of  2
0 variables:
 $ iphone         : num  1 1 1 1 1 41 1 1 1 1 ...
 $ googleandroid  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonedispos   : num  0 0 0 0 0 1 13 0 0 0 ...
 $ iphonedisneg   : num  0 0 0 0 0 3 10 0 0 0 ...
 $ samsunggalaxy  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcphone       : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonedisunc   : num  0 0 0 0 0 4 9 0 0 0 ...
 $ iphoneperpos   : num  0 1 0 1 1 0 5 3 0 0 ...
 $ ios            : num  0 0 0 0 0 6 0 0 0 0 ...
 $ iphoneperneg   : num  0 0 0 0 0 0 4 1 0 0 ...
 $ sonyxperia     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphoneperunc   : num  0 0 0 1 0 0 5 0 0 0 ...
 $ iphonecampos   : num  0 0 0 0 0 1 1 0 0 0 ...
 $ iphonecamneg   : num  0 0 0 0 0 3 1 0 0 0 ...
 $ iphonecamunc   : num  0 0 0 0 0 7 1 0 0 0 ...
 $ htcdisunc      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htccampos      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcperpos      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htccamneg      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonesentiment: num  0 0 0 0 0 4 4 0 0 0 ...
 - attr(*, "spec")=List of 3
  ..$ cols   :List of 59
  .. ..$ iphone          : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ samsunggalaxy  : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ sonyxperia      : list()
```

```
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokialumina    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htcphone       : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ ios            : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ googleandroid  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphonecampos   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungcampos  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonycampos     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiacampos    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htccampos      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphonecamneg   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungcamneg  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonycamneg     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiacamneg    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htccamneg      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphonecamunc   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungcamunc  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonycamunc     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiacamunc    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htccamunc      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphonedispos   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungdispos  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonydispos     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiadispos    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htcdispos      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphonedisneg   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungdisneg  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonydisneg     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiadisneg    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htcdisneg      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphonedisunc   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungdisunc  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonydisunc     : list()
```

```
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ nokiadisunc    : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ htcdisunc      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iphoneperpos   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ samsungperpos  : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ sonyperpos     : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ nokiaperpos    : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ htcperpos      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iphoneperneg   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ samsungperneg  : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ sonyperneg     : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ nokiaperneg    : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ htcperneg      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iphoneperunc   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ samsungperunc  : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ sonyperunc     : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ nokiaperunc    : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ htcperunc      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iosperpos      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ googleperpos   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iosperneg      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ googleperneg   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iosperunc      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ googleperunc   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iphonesentiment: list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  ..$ default: list()
  .. ..- attr(*, "class")= chr  "collector_guess" "collector"
  ..$ skip   : num 1
  ..- attr(*, "class")= chr "col_spec"
```

```
> iphoneRFE$iphonesentiment <- Factor(iphoneRFE$iphonesentiment)
```

==

After preprocessing we have had the following data sets:

- *iphoneDF (this data set retains all of the original features for "out of the box" modeling that follows)*
- *iphoneCOR (this data set doesn't retain the features highly correlated with the dependant)*
- *iphoneNZV (this data set l near zero variance features)*
- *iphoneRFE (this data set retain rfe recommended features)*

==

OUT OF THE BOX MODELING IPHONE

1) Random Forest

```
> set.seed(998)
>
> IntrainingDF<- createDataPartition(iphoneDF$iphonesentiment, p=.70, list=FA
LSE)
> TrainingDF <- iphoneDF[IntrainingDF,]
> TestingDF <- iphoneDF[-IntrainingDF,]
> fitcontrol <- trainControl(method = "repeatedcv", number = 10, repeats = 1)
> rfGrid<- expand.grid(mtry=c(1,2,3,4,5))
> system.time(rfiphoneDF <- train(iphonesentiment~., data = TrainingDF, metho
d = "rf", trControl=fitcontrol, tuneGrid=rfGrid))
user  system elapsed
 667.30   11.04  694.34
> rfiphoneDF
Random Forest

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

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 8173, 8175, 8174, 8176, 8174, 8175, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  1     0.6163219  0.1231823
  2     0.6999928  0.3704923
  3     0.7108918  0.4006543
  4     0.7228908  0.4327856
  5     0.7456802  0.4895226

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 5.

> rfpred <- predict(rfiphoneDF, TestingDF)
>
> rfpred
   [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5
  [41] 0 5 5 0 0 4 5 5 0 5 5 0 5 5 5 5 5 5 5 5 4 5 0 5 5 5 5 0 5 0 5 5 5 0 5
3 5 5 5 5
  [81] 5 5 3 5 5 5 0 5 5 5 5 5 5 5 5 5 2 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5 4 0 0 5
 [121] 5 5 5 5 5 5 5 5 0 5 5 5 5 0 5 5 5 4 5 5 5 5 5 5 5 4 5 5 0 5 5 5 4 0 4
5 5 0 5 5
 [161] 5 5 5 5 5 5 3 4 0 5 5 5 0 4 5 5 5 5 5 5 5 5 5 5 5 5 0 3 0 5 5 0 2 5 5 5
5 0 0 5 0
```

```
 [201] 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 0 5 5 0 5 5 0 5 5 5 5 5 5 5 5 4 5 5
5 5 5 5 5
 [241] 5 5 5 0 5 0 5 5 5 5 5 5 5 5 5 5 5 5 4 5 5 5 5 5 5 3 3 5 5 5 5 0 5 5
5 0 5 5 5
 [281] 5 5 0 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5 0 5 5 5
 [321] 5 3 5 5 5 3 5 5 5 5 5 5 0 5 5 5 4 5 5 5 5 5 5 5 5 0 3 0 5 5 5 5 4 5 5
5 0 5 0 5
 [361] 5 5 5 5 5 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 0 5 0 5 5 5 5 5 5 4 5 5
5 5 5 5 5
 [401] 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 0 5 5 0 5 5 0 5 5 5 0 5 5 5 5 5 5 5
5 5 5 5 5
 [441] 5 5 5 5 5 5 0 5 0 5 4 0 3 5 5 5 4 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 0 3
5 5 5 5 5
 [481] 5 5 0 0 5 5 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5 5 5 5 0 5 5 3 0 5 5
5 3 5 5 5
 [521] 5 0 0 5 5 5 5 0 3 5 5 5 0 5 5 5 5 4 5 5 5 5 5 4 0 0 5 5 0 5 5 0 5 5 5
5 5 5 5 5
 [561] 5 5 5 5 5 0 5 0 5 5 0 5 5 5 4 5 5 5 5 5 5 0 5 5 0 3 4 5 5 5 5 5 0 5 5
5 5 5 4 5
 [601] 5 5 5 0 5 5 5 5 5 0 5 5 5 5 5 4 5 5 5 0 5 5 0 5 5 0 5 5 5 5 0 5 5 5 0
5 0 5 0 0
 [641] 5 4 0 5 0 5 5 5 5 5 0 5 5 5 0 5 5 5 5 5 5 5 5 5 0 4 5 5 0 3 5 0 5 5 5
5 5 5 5 5
 [681] 5 4 5 2 5 5 5 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 0 0 0 5 5 5 5 0 5
5 5 0 5 5
 [721] 5 0 5 5 5 5 5 5 5 5 0 4 5 5 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 3 5 5 0
5 5 5 3 5
 [761] 5 5 5 0 5 0 3 5 5 5 5 5 5 5 0 5 0 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5
 [801] 0 5 5 5 4 5 5 5 5 4 5 5 5 5 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5 5 5 0
5 5 5 5 5
 [841] 5 5 5 4 2 5 5 5 0 5 5 4 5 4 4 5 5 5 5 5 5 5 5 5 4 5 0 5 5 5 5 5 5 5 5
5 5 5 5 3
 [881] 5 5 5 5 5 5 5 5 0 5 5 5 5 0 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 0 5 5 5 5
5 5 0 5 0
 [921] 5 5 3 5 4 5 5 0 5 3 5 5 5 5 5 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
3 5 5 5 5
 [961] 5 0 5 0 5 0 5 5 5 5 5 0 5 5 5 5 0 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5
5 5 3 3 5
 [ reached getOption("max.print") -- omitted 2890 entries ]
Levels: 0 1 2 3 4 5

> table(rfpred, TestingDF$iphonesentiment)

rfpred    0    1    2    3    4    5
     0  385    0    1    3    8    7
     1    0    0    0    0    0    0
     2    0    0   19    0    0    0
     3    1    0    0  145    3    0
     4    0    0    0    3  135    2
     5  202  117  116  205  285 2253
```

2) C5.0

```
> CiphoneDF<- train(iphonesentiment ~ ., data = TrainingDF, method = "C5.0", trcontrol=fitcontrol, tuneLength = 5)

> CiphoneDF
C5.0

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

```
No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 9083, 9083, 9083, 9083, 9083, 9083, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.7651789  0.5484981
  rules  FALSE   10      0.7638379  0.5440713
  rules  FALSE   20      0.7638379  0.5440713
  rules  FALSE   30      0.7638379  0.5440713
  rules  FALSE   40      0.7638379  0.5440713
  rules   TRUE    1      0.7640505  0.5469994
  rules   TRUE   10      0.7655442  0.5470178
  rules   TRUE   20      0.7655442  0.5470178
  rules   TRUE   30      0.7655442  0.5470178
  rules   TRUE   40      0.7655442  0.5470178
  tree   FALSE    1      0.7616676  0.5435295
  tree   FALSE   10      0.7638037  0.5433891
  tree   FALSE   20      0.7638037  0.5433891
  tree   FALSE   30      0.7638037  0.5433891
  tree   FALSE   40      0.7638037  0.5433891
  tree    TRUE    1      0.7615092  0.5432769
  tree    TRUE   10      0.7628382  0.5419166
  tree    TRUE   20      0.7628382  0.5419166
  tree    TRUE   30      0.7628382  0.5419166
  tree    TRUE   40      0.7628382  0.5419166

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 winno
w = TRUE.


> Cpred <- predict(CiphoneDF, TestingDF)
>
> table(Cpred, TestingDF$iphonesentiment)

Cpred    0    1    2    3    4    5
    0  382    0    1   10   13   10
    1    0    0    0    0    0    0
    2    1    0   19    0    0    0
    3    3    1    3  245    2   16
    4    2    0    0    0  136   12
    5  200  116  113  101  280 2224
```

3) SVM

```
> library(e1071)
> SVMiphoneDF <- svm(iphonesentiment ~ . , TrainingDF)

> SVMiphoneDF

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


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

Number of Support Vectors:  4544
```

```
> svmiphone <- train(iphonesentiment ~., data = TrainingDF, method = "svmLine
ar", trControl=fitcontrol, preProcess = c("center", "scale"), tuneLength = 10
)
> svmiphone
Support Vector Machines with Linear Kernel

9083 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: 8175, 8174, 8174, 8173, 8175, 8176, ...
Resampling results:

  Accuracy   Kappa
  0.7014193  0.394746

Tuning parameter 'C' was held constant at a value of 1

> SVMpred <- predict(SVMiphoneDF, TestingDF)
>
> table(SVMpred, TestingDF$iphonesentiment)

SVMpred    0    1    2    3    4    5
      0  361    1    6   27   21   59
      1    0    0    0    0    0    0
      2    0    0    2    0    0    0
      3    0    1   15  109    2    3
      4    1    0    0    1  124    1
      5  226  115  113  219  284 2199
```

4) KKNN

```
>set.seed(998)
> kknniphoneDF=train.kknn(iphonesentiment ~ ., data = TrainingDF, kmax = 100,
kernel = c("optimal","rectangular", "inv", "gaussian", "triangular"), scale =
TRUE)
> kknniphoneDF


Call:
train.kknn(formula = iphonesentiment ~ ., data = TrainingDF,     kmax = 100,
kernel = c("optimal", "rectangular", "inv", "gaussian",         "triangular")
, scale = TRUE)

Type of response variable: nominal
Minimal misclassification: 0.5698558
Best kernel: inv
Best k: 86

> knnpred <- predict(kknniphoneDF, TestingDF)
> knnpred


[1] 5 5 5 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 5 5 0 0 0 0 0 0 0 0 0 0 5 5 5 5
0 0 3
  [41] 0 5 0 0 0 5 0 0 0 0 0 0 5 0 0 5 0 5 0 5 4 5 0 0 5 0 0 0 5 0 0 5 0 5 0
3 0 0 5 5
  [81] 0 0 3 0 0 0 0 5 0 0 0 5 0 5 0 5 2 0 5 0 0 0 0 0 5 5 5 0 0 0 0 0 0 5 0
0 4 0 0 0
 [121] 0 5 3 5 5 5 0 0 0 0 5 5 0 0 0 0 0 4 0 5 0 0 5 0 0 4 0 0 0 0 0 5 4 0 4
5 3 0 0 0
```

```
 [161] 0 5 0 5 0 0 3 4 0 0 0 0 0 4 5 5 0 0 0 0 0 0 5 5 0 0 3 0 5 0 0 2 0 5 0
0 0 0 0 0
 [201] 5 0 5 0 0 0 0 5 5 0 0 0 0 0 5 5 0 0 5 0 5 0 0 5 3 0 0 0 0 5 0 5 4 0 0
0 5 0 0 0
 [241] 0 0 0 0 5 0 0 5 0 0 0 0 5 0 5 5 0 5 0 4 0 5 0 5 5 0 3 3 5 5 0 0 0 0 0
0 0 0 5 5
 [281] 5 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 5 0 0 0 3 5 5 5 0 0 0 5 5 0 0 0 0 0 5
5 0 0 0 5
 [321] 5 3 0 0 5 3 5 0 0 3 3 0 0 0 0 0 4 0 5 0 5 5 5 5 5 0 3 0 0 0 0 0 4 5 5
5 0 5 0 0
 [361] 4 3 0 5 5 3 1 5 0 0 0 5 5 0 0 0 5 0 0 5 0 0 0 0 0 0 3 0 0 0 5 5 4 0 0
0 0 0 3 5
 [401] 3 0 0 5 0 0 5 5 5 0 5 5 0 5 5 0 0 0 3 0 0 0 0 0 5 0 0 0 5 5 5 5 0 0 0
0 3 0 5 5
 [441] 5 5 0 0 5 5 0 0 0 0 4 0 3 0 0 5 4 0 0 3 0 5 5 0 3 0 0 0 0 5 5 0 0 0 3
5 5 0 5 0
 [481] 5 0 0 0 5 0 0 5 5 5 0 0 5 0 0 5 0 5 4 5 3 5 0 0 0 0 0 5 0 0 0 3 0 0 0
0 3 5 0 5
 [521] 3 0 0 0 5 0 0 0 3 0 5 5 0 0 0 0 0 4 0 0 5 0 5 4 0 0 0 0 0 0 0 0 0 0 0
0 0 0 5 5
 [561] 0 0 5 0 0 5 0 0 0 0 4 0 0 0 4 0 5 5 5 5 0 0 0 0 0 3 4 5 0 0 5 5 0 0 5
5 0 0 4 0
 [601] 0 5 0 0 5 0 0 5 5 0 0 0 5 0 0 4 5 5 0 0 0 0 0 0 5 0 0 0 3 0 0 5 0 0 0
5 0 0 0 0
 [641] 5 4 0 5 0 0 0 0 5 0 0 5 0 0 0 0 0 0 5 3 5 0 0 0 0 4 0 5 0 3 0 0 0 0 0
5 3 0 0 0
 [681] 0 4 5 2 0 5 0 0 5 5 0 0 0 0 0 5 0 0 5 5 0 5 5 5 0 5 0 0 0 0 5 0 0 0 3
5 5 0 0 0
 [721] 0 0 5 5 5 5 0 0 5 0 0 4 5 0 0 0 0 0 5 0 0 3 0 0 5 0 5 0 0 5 5 3 0 0 0
0 5 0 3 5
 [761] 5 5 5 0 3 0 3 5 0 5 0 5 5 0 0 0 0 0 0 0 0 5 0 0 0 0 0 5 5 5 0 0 5 5 0
0 0 5 0 0
 [801] 0 5 5 0 4 5 0 0 5 4 5 0 0 5 5 0 0 5 0 5 0 0 4 0 0 3 0 0 0 0 0 5 0 0 0
5 0 5 5 0
 [841] 0 5 0 0 2 5 5 5 0 5 0 4 0 4 4 5 5 0 0 0 5 0 0 5 4 0 0 5 0 0 0 0 5 0 0
0 5 0 0 3
 [881] 5 0 0 0 5 5 5 0 0 0 0 0 0 0 4 0 0 5 5 0 3 3 0 0 5 5 5 0 0 0 0 0 3 0 5
0 0 0 0 0
 [921] 0 5 0 5 4 5 5 0 5 3 0 0 5 0 5 0 3 0 3 0 0 5 0 0 0 0 0 0 0 5 5 5 0 0 0
3 0 0 0 5
 [961] 0 0 5 0 5 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 5 0 0 0 0 5 0 5 5 5 0 5
0 0 3 3 0
 [ reached getOption("max.print") -- omitted 2890 entries ]
Levels: 0 1 2 3 4 5
```

POSTRESAMPLE ()

Random Forest

```
> postResample(rfpred, TestingDF$iphonesentiment)
 Accuracy     Kappa 
0.7550129 0.5115682 
> summary(rfpred)
   0    1    2    3    4    5 
 404    0   19  149  140 3178 
```

C5.0

```
> postResample(Cpred, TestingDF$iphonesentiment)
 Accuracy     Kappa 
0.7727506 0.5625596 
> summary(Cpred)
   0    1    2    3    4    5 
```

```
 416    0   20  270  150 3034
```

SVM

```
> postResample(SVMpred, TestingDF$iphonesentiment)
 Accuracy     Kappa 
0.7185090 0.4404634 
> summary(SVMpred)
   0    1    2    3    4    5 
 475    0    2  130  127 3156 
```

KKNN

```
> postResample(kknnpred, TestingDF$iphonesentiment)
 Accuracy     Kappa 
0.4514139 0.2440512 
> summary(kknnpred)
   0    1    2    3    4    5 
2288    7   20  255  150 1170 
```

==

After exploring the results from several methods in our OUT OF THE BOX MODELING. The PostResample() function showed that the best classifier performance was the C5.0 model with an accuracy value of 0.7727506 and kappa value of 0.5625596.

==

CONFUSION MATRIX

Because we have found that some models had very similar accuracy and kappa, we have explored additional metrics available from the confusion matrix:

Random Forest

```
> cmRF <- confusionMatrix(rfpred, TestingDF$iphonesentiment)
> cmRF
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5
         0  385    0    1    3    8    7
         1    0    0    0    0    0    0
         2    0    0   19    0    0    0
         3    1    0    0  145    3    0
         4    0    0    0    3  135    2
         5  202  117  116  205  285 2253

Overall Statistics
                                          
               Accuracy : 0.755           
                 95% CI : (0.7412, 0.7685)
    No Information Rate : 0.5815          
    P-Value [Acc > NIR] : < 2.2e-16       
                                          
                  Kappa : 0.5116          
                                          
 Mcnemar's Test P-Value : NA              
```

```
Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity           0.65476  0.00000 0.139706  0.40730  0.31323   0.9960
Specificity           0.99425  1.00000 1.000000  0.99887  0.99855   0.4318
Pos Pred Value        0.95297      NaN 1.000000  0.97315  0.96429   0.7089
Neg Pred Value        0.94177  0.96992 0.969775  0.94360  0.92107   0.9874
Prevalence            0.15116  0.03008 0.034961  0.09152  0.11080   0.5815
Detection Rate        0.09897  0.00000 0.004884  0.03728  0.03470   0.5792
Detection Prevalence  0.10386  0.00000 0.004884  0.03830  0.03599   0.8170
Balanced Accuracy     0.82450  0.50000 0.569853  0.70309  0.65589   0.7139
```

C5.0

```
> cmC <- confusionMatrix(Cpred, TestingDF$iphonesentiment)
>
> cmC
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5
         0  382    0    1   10   13   10
         1    0    0    0    0    0    0
         2    1    0   19    0    0    0
         3    3    1    3  245    2   16
         4    2    0    0    0  136   12
         5  200  116  113  101  280 2224

Overall Statistics

               Accuracy : 0.7728
                 95% CI : (0.7593, 0.7858)
    No Information Rate : 0.5815
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5626

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity            0.6497  0.00000 0.139706  0.68820  0.31555   0.9832
Specificity            0.9897  1.00000 0.999734  0.99293  0.99595   0.5025
Pos Pred Value         0.9183      NaN 0.950000  0.90741  0.90667   0.7330
Neg Pred Value         0.9407  0.96992 0.969767  0.96934  0.92112   0.9556
Prevalence             0.1512  0.03008 0.034961  0.09152  0.11080   0.5815
Detection Rate         0.0982  0.00000 0.004884  0.06298  0.03496   0.5717
Detection Prevalence   0.1069  0.00000 0.005141  0.06941  0.03856   0.7799
Balanced Accuracy      0.8197  0.50000 0.569720  0.84056  0.65575   0.7428
```

SVM

```
> cmSVM <- confusionMatrix(SVMpred, TestingDF$iphonesentiment)
>
> cmSVM
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5
         0  361    1    6   27   21   59
         1    0    0    0    0    0    0
```

```
         2    0    0    2    0    0    0
         3    0    1   15  109    2    3
         4    1    0    0    1  124    1
         5  226  115  113  219  284 2199

Overall Statistics

               Accuracy : 0.7185
                 95% CI : (0.7041, 0.7326)
    No Information Rate : 0.5815
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4405

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1  Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity            0.6139  0.00000 0.0147059  0.30618  0.28770   0.9721
Specificity            0.9655  1.00000 1.0000000  0.99406  0.99913   0.4122
Pos Pred Value         0.7600      NaN 1.0000000  0.83846  0.97638   0.6968
Neg Pred Value         0.9335  0.96992 0.9655350  0.93431  0.91842   0.9142
Prevalence             0.1512  0.03008 0.0349614  0.09152  0.11080   0.5815
Detection Rate         0.0928  0.00000 0.0005141  0.02802  0.03188   0.5653
Detection Prevalence   0.1221  0.00000 0.0005141  0.03342  0.03265   0.8113
Balanced Accuracy      0.7897  0.50000 0.5073529  0.65012  0.64342   0.6922
```

KKNN

```
> cmkknn <- confusionMatrix(kknnpred, TestingDF$iphonesentiment)
>
> cmkknn
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5
         0  515   72   68   75  178 1380
         1    0    1    0    0    0    6
         2    0    0   19    0    0    1
         3    3    0    1  244    1    6
         4    0    1    1    0  128   20
         5   70   43   47   37  124  849

Overall Statistics

               Accuracy : 0.4514
                 95% CI : (0.4357, 0.4672)
    No Information Rate : 0.5815
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2441

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0  Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity            0.8759 0.0085470 0.139706  0.68539  0.29698   0.3753
Specificity            0.4631 0.9984098 0.999734  0.99689  0.99364   0.8028
Pos Pred Value         0.2251 0.1428571 0.950000  0.95686  0.85333   0.7256
Neg Pred Value         0.9544 0.9701262 0.969767  0.96919  0.91898   0.4805
Prevalence             0.1512 0.0300771 0.034961  0.09152  0.11080   0.5815
```

```
Detection Rate         0.1324 0.0002571 0.004884  0.06272  0.03290   0.2183
Detection Prevalence   0.5882 0.0017995 0.005141  0.06555  0.03856   0.3008
Balanced Accuracy      0.6695 0.5034784 0.569720  0.84114  0.64531   0.5891
```

MODELING IPHONE SMALL MATRIX FEATURE SELECTION DATASET

I have chosen to go further with my modeling for Dataset "iphoneRFE" with the three algorithms C5.0, Random Forest and SVM as they showed very similar accuracy and kappa in my "Out of the Box work".

C5.0

```
> CiphoneRFE<- train(iphonesentiment ~ ., data = TrainingRFE, method = "C5.0",
trcontrol=fitcontrol, tuneLength = 5)
>
> CiphoneRFE
C5.0

9082 samples
  19 predictor
   6 classes: '0', '1', '2', '3', '4', '5'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 9082, 9082, 9082, 9082, 9082, 9082, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.7640256  0.5418908
  rules  FALSE   10      0.7613726  0.5353221
  rules  FALSE   20      0.7613726  0.5353221
  rules  FALSE   30      0.7613726  0.5353221
  rules  FALSE   40      0.7613726  0.5353221
  rules   TRUE    1      0.7634423  0.5411514
  rules   TRUE   10      0.7613585  0.5356218
  rules   TRUE   20      0.7613585  0.5356218
  rules   TRUE   30      0.7613585  0.5356218
  rules   TRUE   40      0.7613585  0.5356218
  tree   FALSE    1      0.7617028  0.5385339
  tree   FALSE   10      0.7615666  0.5364469
  tree   FALSE   20      0.7615666  0.5364469
  tree   FALSE   30      0.7615666  0.5364469
  tree   FALSE   40      0.7615666  0.5364469
  tree    TRUE    1      0.7619427  0.5389479
  tree    TRUE   10      0.7607192  0.5346121
  tree    TRUE   20      0.7607192  0.5346121
  tree    TRUE   30      0.7607192  0.5346121
  tree    TRUE   40      0.7607192  0.5346121

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.

> CpredRFE <- predict(CiphoneRFE, TestingRFE)
>
> cmC_RFE <- confusionMatrix(CpredRFE, TestingRFE$iphonesentiment)
>
> cmC_RFE
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5
         0  391    0    2    4    2    3
         1    0    0    0    0    0    0
         2    2    0   19    0    0    0
```

```
          3    3    0    1  227    6    9
          4    1    0    0    0  142    6
          5  199  121  111  117  289 2236

Overall Statistics

               Accuracy : 0.7749
                 95% CI : (0.7614, 0.7879)
    No Information Rate : 0.5793
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5641

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity            0.6560   0.0000 0.142857  0.65230  0.32346   0.9920
Specificity            0.9967   1.0000 0.999468  0.99464  0.99797   0.4887
Pos Pred Value         0.9726      NaN 0.904762  0.92276  0.95302   0.7276
Neg Pred Value         0.9412   0.9689 0.970543  0.96680  0.92063   0.9780
Prevalence             0.1532   0.0311 0.034181  0.08944  0.11282   0.5793
Detection Rate         0.1005   0.0000 0.004883  0.05834  0.03649   0.5747
Detection Prevalence   0.1033   0.0000 0.005397  0.06322  0.03829   0.7898
Balanced Accuracy      0.8264   0.5000 0.571162  0.82347  0.66072   0.7404
```

Random Forest

```
> library(caret)
> set.seed(998)
> IntrainingRFE<- createDataPartition(iphoneRFE$iphonesentiment, p=.70, list=
FALSE)
> TrainingRFE <- iphoneRFE[IntrainingRFE,]
> TestingRFE <- iphoneRFE[-IntrainingRFE,]
> fitcontrol <- trainControl(method = "repeatedcv", number = 10, repeats = 1)
> rfGrid<- expand.grid(mtry=c(1,2,3,4,5))
> system.time(rfiphoneRFE <- train(iphonesentiment~., data = TrainingRFE, met
hod = "rf", trControl=fitcontrol, tuneGrid=rfGrid))
> rfiphoneRFE
Random Forest

9082 samples
  19 predictor
   6 classes: '0', '1', '2', '3', '4', '5'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 8174, 8173, 8174, 8173, 8175, 8174, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  1     0.6783745  0.3052354
  2     0.7392665  0.4728598
  3     0.7686641  0.5471204
  4     0.7728486  0.5568518
  5     0.7731793  0.5585841

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 5.

> rfpredRFE <- predict(rfiphoneRFE, TestingRFE)
```

```
> cmRF_RFE <- confusionMatrix(rfpredRFE, TestingRFE$iphonesentiment)
> cmRF_RFE
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5
         0  402    0    1    2    4    7
         1    0    0    0    0    0    1
         2    0    1   19    0    0    1
         3    0    0    0  229    1    6
         4    1    0    1    0  151    3
         5  193  120  112  117  283 2236

Overall Statistics

               Accuracy : 0.7805
                 95% CI : (0.7672, 0.7934)
    No Information Rate : 0.5793
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5761

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity            0.6745 0.000000 0.142857  0.65805  0.34396   0.9920
Specificity            0.9958 0.999735 0.999468  0.99802  0.99855   0.4960
Pos Pred Value         0.9663 0.000000 0.904762  0.97034  0.96795   0.7305
Neg Pred Value         0.9442 0.968895 0.970543  0.96744  0.92289   0.9783
Prevalence             0.1532 0.031097 0.034181  0.08944  0.11282   0.5793
Detection Rate         0.1033 0.000000 0.004883  0.05885  0.03881   0.5747
Detection Prevalence   0.1069 0.000257 0.005397  0.06065  0.04009   0.7867
Balanced Accuracy      0.8351 0.499867 0.571162  0.82804  0.67126   0.7440
```

SVM

```
> svmiphoneRFE <- train(iphonesentiment ~., data = TrainingRFE, method = "svm
Linear", trControl=fitcontrol, preProcess = c("center", "scale"), tuneLength
= 10)
>
> svmiphoneRFE
Support Vector Machines with Linear Kernel

9082 samples
  19 predictor
   6 classes: '0', '1', '2', '3', '4', '5'

Pre-processing: centered (19), scaled (19)
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 8174, 8174, 8174, 8173, 8173, 8173, ...
Resampling results:

  Accuracy   Kappa
  0.7043603  0.4008678

Tuning parameter 'C' was held constant at a value of 1

Tuning parameter 'C' was held constant at a value of 1

> SVMpredRFE <- predict(svmiphoneRFE, TestingRFE)
> cmSVMpredRFE <- confusionMatrix(SVMpredRFE, TestingRFE$iphonesentiment)
```

```
> cmSVMpredRFE
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5
         0  396    4    2   23   16   44
         1    0    0    0    0    0    0
         2    0    0    0    0    2    1
         3    4    0   19   94    1    7
         4    0    0    1    0   88    4
         5  196  117  111  231  332 2198

Overall Statistics

               Accuracy : 0.7134
                 95% CI : (0.6989, 0.7276)
    No Information Rate : 0.5793
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4282

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity            0.6644   0.0000 0.000000  0.27011  0.20046   0.9752
Specificity            0.9730   1.0000 0.999202  0.99125  0.99855   0.3971
Pos Pred Value         0.8165      NaN 0.000000  0.75200  0.94624   0.6901
Neg Pred Value         0.9413   0.9689 0.965792  0.93255  0.90758   0.9207
Prevalence             0.1532   0.0311 0.034181  0.08944  0.11282   0.5793
Detection Rate         0.1018   0.0000 0.000000  0.02416  0.02262   0.5649
Detection Prevalence   0.1246   0.0000 0.000771  0.03213  0.02390   0.8186
Balanced Accuracy      0.8187   0.5000 0.499601  0.63068  0.59950   0.6861
```

==

*From the above data we can see that our Random Forest classifier correctly identified class **5** 99 % of the time and for Classes **0** and **3** 67% and 65% of the time.*

*Further, when we shouldn't have predicted class **5,** we didn't for 49% of examples. We can contrast this to classes **4, 2, 1**: our specificity (true negative) is over 99% but our sensitivity (true positive) is around 34%, 14% and 0% which make us think that we do a poor job of positively identifying items of these classes. But the positive predictive value is of over 90%: despite our classifier only being able to positively identify objects 34% and 14% of the time there's over a 90% chance that, when it does, such a classification is correct.*

==

MODELING WITH ENGINEERING THE DEPENDANT VARIABLE

Random Forest

```
> iphoneRC <- iphoneDF
> library(dplyr)
> iphoneRC$iphonesentiment <- recode(iphoneRC$iphonesentiment, '0' = 1, '1' = 
1, '2' = 2, '3' = 3, '4' = 4, '5' = 4)
> set.seed(998)
```

```
> rfpredRC <- predict(rfiphoneRC, TestingRC)
> rfCMRC <- confusionMatrix(rfpredRC, TestingRC$iphonesentiment)
> rfCMRC
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3    4
         1  362    1    4   13
         2    0   17    0    0
         3    2    0  141    1
         4  341  118  211 2679

Overall Statistics

               Accuracy : 0.8224
                 95% CI : (0.81, 0.8343)
    No Information Rate : 0.6923
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5359

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity           0.51348  0.12500  0.39607   0.9948
Specificity           0.99435  1.00000  0.99915   0.4403
Pos Pred Value        0.95263  1.00000  0.97917   0.7999
Neg Pred Value        0.90228  0.96927  0.94261   0.9741
Prevalence            0.18123  0.03496  0.09152   0.6923
Detection Rate        0.09306  0.00437  0.03625   0.6887
Detection Prevalence  0.09769  0.00437  0.03702   0.8609
Balanced Accuracy     0.75391  0.56250  0.69761   0.7175
```

C5.0

```
> CiphoneRC<- train(iphonesentiment ~ ., data = TrainingRC, method = "C5.0",
+                    trcontrol=fitcontrol, tuneLength = 5)
> CiphoneRC
C5.0

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

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 9083, 9083, 9083, 9083, 9083, 9083, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.8457616  0.6174733
  rules  FALSE   10      0.8410146  0.6092272
  rules  FALSE   20      0.8410146  0.6092272
  rules  FALSE   30      0.8410146  0.6092272
  rules  FALSE   40      0.8410146  0.6092272
  rules   TRUE    1      0.8454930  0.6177138
  rules   TRUE   10      0.8415749  0.6092283
  rules   TRUE   20      0.8415749  0.6092283
  rules   TRUE   30      0.8415749  0.6092283
  rules   TRUE   40      0.8415749  0.6092283
  tree   FALSE    1      0.8440963  0.6145003
  tree   FALSE   10      0.8403471  0.6085956
```

```
  tree    FALSE   20       0.8403471  0.6085956
  tree    FALSE   30       0.8403471  0.6085956
  tree    FALSE   40       0.8403471  0.6085956
  tree     TRUE    1       0.8436443  0.6141695
  tree     TRUE   10       0.8395594  0.6058349
  tree     TRUE   20       0.8395594  0.6058349
  tree     TRUE   30       0.8395594  0.6058349
  tree     TRUE   40       0.8395594  0.6058349

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.
> CpredRC <- predict(CiphoneRC, TestingRC)

> summary(CpredRC)
   1    2    3    4 
 377   17  268 3228 
```

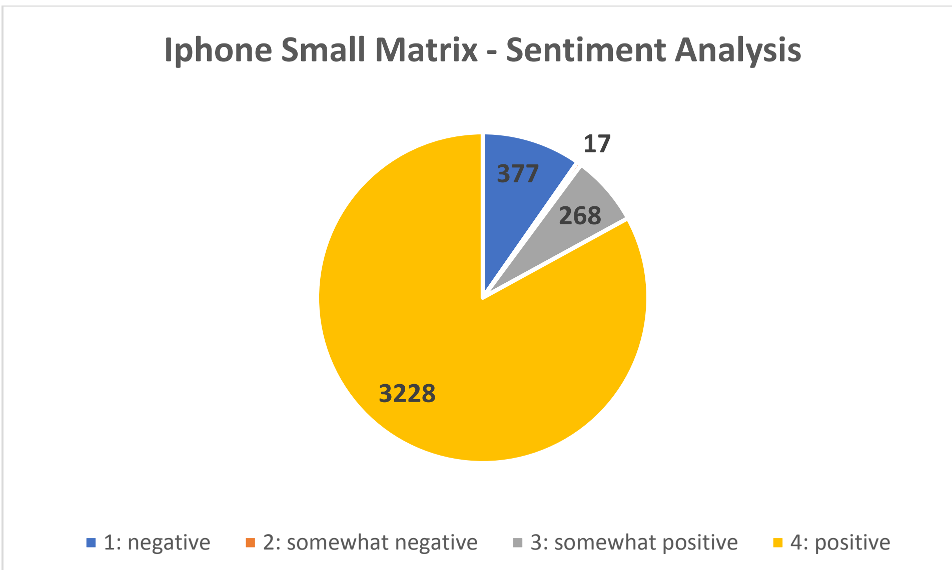


Figure 1: Iphone Small Matrix - Sentiment Analysis

```
> CcmRC <- confusionMatrix(CpredRC, TestingRC$iphonesentiment)
> CcmRC
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3    4
         1  361    0    5   11
         2    0   17    0    0
         3    6    3  243   16
         4  338  116  108 2666

Overall Statistics
                                          
               Accuracy : 0.845           
                 95% CI : (0.8332, 0.8562)
    No Information Rate : 0.6923          
    P-Value [Acc > NIR] : < 2.2e-16       
```

```
                  Kappa : 0.6139

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity           0.51206  0.12500  0.68258   0.9900
Specificity           0.99498  1.00000  0.99293   0.5305
Pos Pred Value        0.95756  1.00000  0.90672   0.8259
Neg Pred Value        0.90208  0.96927  0.96880   0.9592
Prevalence            0.18123  0.03496  0.09152   0.6923
Detection Rate        0.09280  0.00437  0.06247   0.6853
Detection Prevalence  0.09692  0.00437  0.06889   0.8298
Balanced Accuracy     0.75352  0.56250  0.83776   0.7602
```

==

We can clearly see that the engineering of the dependent variable showed an improvement in the skill of our best models Random Forest (Accuracy: 0.8224, kappa: 0.5359) and C5.0 (Accuracy: 0.8457616, kappa: 0.6174733).

==

PRINCIPAL COMPONENT ANALYSIS

```
> preprocessParams <- preProcess(Training[,-59], method=c("center", "scale", 
"pca"), thresh = 0.95)
>
> print(preprocessParams)
Created from 9083 samples and 58 variables

Pre-processing:
  - centered (58)
  - ignored (0)
  - principal component signal extraction (58)
  - scaled (58)

PCA needed 25 components to capture 95 percent of the variance
```

==

In order to capture 95% of the variance, the output showed us that we need 25 components. Also, whenever we lower the variance threshold, the number of components gets lower as well.

==

```
> train.pca <- predict(preprocessParams, Training[,-59])

> train.pca$iphonesentiment <- Training$iphonesentiment

> Testing <- iphoneDF[-IntrainingDF,]

> test.pca <- predict(preprocessParams, Testing[,-59])

> test.pca$iphonesentiment <- Testing$iphonesentiment

> str(train.pca)
'data.frame':   9083 obs. of  26 variables:
 $ PC1            : num  -0.69 -0.69 -0.628 0.706 1.751 ...
```

```
 $ PC2             : num  0.0235 0.0235 0.0151 -0.2774 -0.423 ...
 $ PC3             : num  -0.329 -0.329 -0.206 3.596 5.066 ...
 $ PC4             : num  0.454 0.454 0.352 -2.755 -3.92 ...
 $ PC5             : num  -0.1527 -0.1527 -0.0742 0.7343 3.3922 ...
 $ PC6             : num  0.195 0.195 0.271 -8.962 1.551 ...
 $ PC7             : num  0.0924 0.0924 0.0733 0.7787 2.2647 ...
 $ PC8             : num  -0.0993 -0.0993 -0.026 -5.6761 1.6085 ...
 $ PC9             : num  0.0388 0.0388 0.0297 1.1843 -0.7097 ...
 $ PC10            : num  -0.0218 -0.0218 0.0107 -0.9653 1.0856 ...
 $ PC11            : num  -0.28 -0.28 -0.309 0.959 0.197 ...
 $ PC12            : num  0.0175 0.0175 0.0931 -0.7431 -0.8345 ...
 $ PC13            : num  -0.1231 -0.1231 -0.1319 0.0829 0.1144 ...
 $ PC14            : num  0.0378 0.0378 -0.0418 -0.0613 0.0206 ...
 $ PC15            : num  -0.1186 -0.1186 0.0542 0.3432 -0.2141 ...
 $ PC16            : num  0.0657 0.0657 -0.0475 -0.4792 -0.1262 ...
 $ PC17            : num  -0.00887 -0.00887 -0.03655 0.03448 0.01625 ...
 $ PC18            : num  0.00839 0.00839 -0.08041 -1.10061 0.05058 ...
 $ PC19            : num  -0.0192 -0.0192 0.0207 0.1547 0.0316 ...
 $ PC20            : num  0.075 0.075 0.0219 -0.2095 0.107 ...
 $ PC21            : num  0.0674 0.0674 0.0382 -0.0573 0.0388 ...
 $ PC22            : num  0.0549 0.0549 0.1126 0.0906 -0.3344 ...
 $ PC23            : num  0.03201 0.03201 0.01504 -0.00713 0.0731 ...
 $ PC24            : num  -0.0255 -0.0255 -0.064 -0.1091 0.1465 ...
 $ PC25            : num  0.00465 0.00465 -0.01935 -0.06313 0.68997 ...
 $ iphonesentiment: Factor w/ 6 levels "0","1","2","3",..: 1 1 1 5 5 1 4 1 1
1 ...
> str(test.pca)
'data.frame':	3890 obs. of  26 variables:
 $ PC1             : num  -0.628 -0.529 -0.427 -0.69 -0.69 ...
 $ PC2             : num  0.01507 -0.00242 -0.01366 0.02346 0.02346 ...
 $ PC3             : num  -0.2061 -0.0157 0.1513 -0.3286 -0.3286 ...
 $ PC4             : num  0.3522 0.1979 0.0539 0.4539 0.4539 ...
 $ PC5             : num  -0.0742 0.0891 0.1995 -0.1527 -0.1527 ...
 $ PC6             : num  0.271 0.428 0.497 0.195 0.195 ...
 $ PC7             : num  0.0733 0.1225 0.0433 0.0924 0.0924 ...
 $ PC8             : num  -0.026 0.0297 0.2215 -0.0993 -0.0993 ...
 $ PC9             : num  0.02972 0.0151 -0.00372 0.03878 0.03878 ...
 $ PC10            : num  0.0107 0.0129 0.028 -0.0218 -0.0218 ...
 $ PC11            : num  -0.309 -0.289 -0.352 -0.28 -0.28 ...
 $ PC12            : num  0.0931 0.1205 0.2278 0.0175 0.0175 ...
 $ PC13            : num  -0.132 -0.176 -0.17 -0.123 -0.123 ...
 $ PC14            : num  -0.0418 -0.1649 -0.2705 0.0378 0.0378 ...
 $ PC15            : num  0.0542 0.3618 0.5641 -0.1186 -0.1186 ...
 $ PC16            : num  -0.0475 -0.2411 -0.3732 0.0657 0.0657 ...
 $ PC17            : num  -0.03655 -0.03939 -0.11511 -0.00887 -0.00887 ...
 $ PC18            : num  -0.08041 -0.10486 -0.31249 0.00839 0.00839 ...
 $ PC19            : num  0.02073 0.00307 0.06337 -0.01918 -0.01918 ...
 $ PC20            : num  0.0219 0.0545 -0.1074 0.075 0.075 ...
 $ PC21            : num  0.03824 0.04054 -0.00169 0.06738 0.06738 ...
 $ PC22            : num  0.1126 -0.0289 0.2807 0.0549 0.0549 ...
 $ PC23            : num  0.015 0.0162 -0.0216 0.032 0.032 ...
 $ PC24            : num  -0.064 0.0195 -0.1541 -0.0255 -0.0255 ...
 $ PC25            : num  -0.01935 0.25409 -0.18919 0.00465 0.00465 ...
 $ iphonesentiment: Factor w/ 6 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1
1 ...
```

MODELING with TRAIN.PCA and TEST.PCA

C5.0

```
> CiphonePCA<- train(iphonesentiment ~ ., data = train.pca, method = "C5.0",
trcontrol=fitcontrol, tuneLength = 5)
```

```
> CiphonePCA
C5.0

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

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 9083, 9083, 9083, 9083, 9083, 9083, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.7495614  0.5178470
  rules  FALSE   10      0.7549190  0.5252076
  rules  FALSE   20      0.7549190  0.5252076
  rules  FALSE   30      0.7549190  0.5252076
  rules  FALSE   40      0.7549190  0.5252076
  rules   TRUE    1      0.7496444  0.5179954
  rules   TRUE   10      0.7547289  0.5249077
  rules   TRUE   20      0.7547289  0.5249077
  rules   TRUE   30      0.7547289  0.5249077
  rules   TRUE   40      0.7547289  0.5249077
  tree   FALSE    1      0.7466616  0.5143666
  tree   FALSE   10      0.7550735  0.5234409
  tree   FALSE   20      0.7550735  0.5234409
  tree   FALSE   30      0.7550735  0.5234409
  tree   FALSE   40      0.7550735  0.5234409
  tree    TRUE    1      0.7464688  0.5141908
  tree    TRUE   10      0.7556588  0.5245499
  tree    TRUE   20      0.7556588  0.5245499
  tree    TRUE   30      0.7556588  0.5245499
  tree    TRUE   40      0.7556588  0.5245499

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
= TRUE.

> CpredPCA <- predict(CiphonePCA, test.pca)
>
> CcmPCA <- confusionMatrix(CpredPCA, test.pca$iphonesentiment)
> CcmPCA
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5
         0  391    0    4   18   17   21
         1    0    0    0    0    0    0
         2    0    0   16    0    0    0
         3    0    0    4  232    1    9
         4    1    2    0    0  127    6
         5  196  115  112  106  286 2226

Overall Statistics

               Accuracy : 0.7692
                 95% CI : (0.7556, 0.7823)
    No Information Rate : 0.5815
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5544
```

```
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity            0.6650  0.00000 0.117647  0.65169  0.29466   0.9841
Specificity            0.9818  1.00000 1.000000  0.99604  0.99740   0.4994
Pos Pred Value         0.8670      NaN 1.000000  0.94309  0.93382   0.7320
Neg Pred Value         0.9427  0.96992 0.969024  0.96597  0.91902   0.9576
Prevalence             0.1512  0.03008 0.034961  0.09152  0.11080   0.5815
Detection Rate         0.1005  0.00000 0.004113  0.05964  0.03265   0.5722
Detection Prevalence   0.1159  0.00000 0.004113  0.06324  0.03496   0.7817
Balanced Accuracy      0.8234  0.50000 0.558824  0.82386  0.64603   0.7417
```

Random Forest

```
> rfiphonePCA<- train(iphonesentiment~., data = train.pca, method = "rf", trC
ontrol=fitcontrol, tuneGrid=rfGrid)
>
> rfiphonePCA
Random Forest

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

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 8175, 8175, 8175, 8173, 8176, 8176, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  1     0.7576806  0.5336412
  2     0.7582312  0.5344746
  3     0.7588916  0.5363796
  4     0.7592221  0.5373672
  5     0.7577912  0.5347936

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 4.

> rfpredPCA <- predict(rfiphonePCA, test.pca)

> rfCMPCA <- confusionMatrix(rfpredPCA, test.pca$iphonesentiment)

> rfCMPCA
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5
         0  386    1    6    7   14   31
         1    0    1    0    0    0    7
         2    1    0   17    0    1    2
         3    0    0    0  243    2    4
         4    7    1    1    4  136   25
         5  194  114  112  102  278 2193

Overall Statistics

               Accuracy : 0.765
                 95% CI : (0.7514, 0.7783)
    No Information Rate : 0.5815
    P-Value [Acc > NIR] : < 2.2e-16
```

```
                 Kappa : 0.5518

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0  Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity           0.65646 0.0085470 0.125000  0.68258  0.31555   0.9695
Specificity           0.98213 0.9981447 0.998934  0.99830  0.98901   0.5086
Pos Pred Value        0.86742 0.1250000 0.809524  0.97590  0.78161   0.7327
Neg Pred Value        0.94136 0.9701185 0.969243  0.96896  0.92061   0.9231
Prevalence            0.15116 0.0300771 0.034961  0.09152  0.11080   0.5815
Detection Rate        0.09923 0.0002571 0.004370  0.06247  0.03496   0.5638
Detection Prevalence  0.11440 0.0020566 0.005398  0.06401  0.04473   0.7694
Balanced Accuracy     0.81930 0.5033459 0.561967  0.84044  0.65228   0.7390
```

III. IPHONE LARGE MATRIX SENTIMENT ANALYSIS

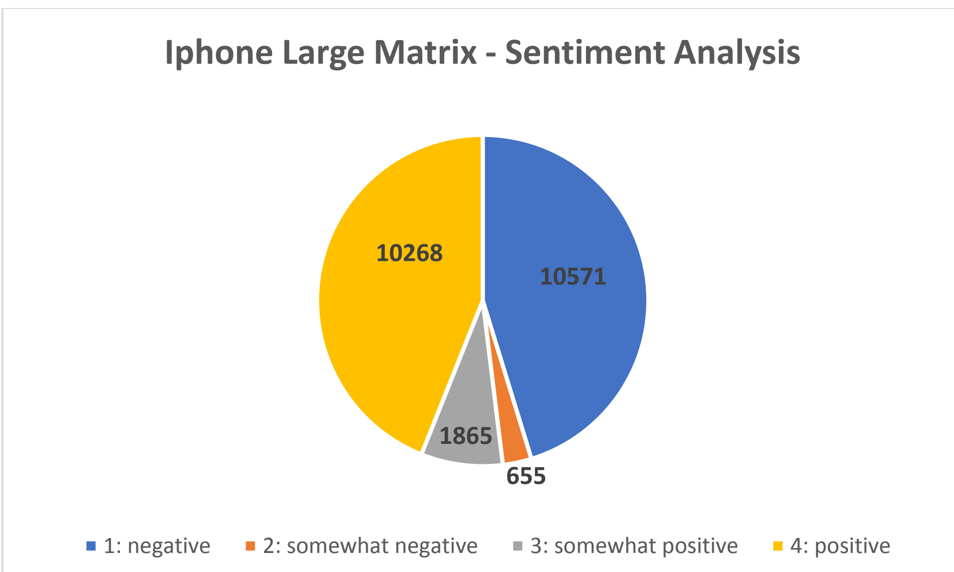
===

So far, the best modeling was the one using the ***C5.0 algorithm*** *with the engineering dependent variable, because it has the highest values for both Accuracy with a value of 87% and Kappa with a value of 67%.*

===

```
> library(caret)

> iphoneLargeMatrix <- read_csv("~/UT Data Analytics Course/Course 4/Task 3 -
Predict Sentiment/iphoneLargeMatrix.csv")

> set.seed(998)

> LargeMatrixPred <- predict(CiphoneRC, iphoneLargeMatrix)
> LargeMatrixPred
   [1] 4 2 4 4 3 4 4 1 3 1 4 4 4 4 1 1 3 1 4 4 4 1 1 4 1 4 1 4 1 4 3 3 4 1 4
1 1 1 1 4
  [41] 1 4 4 1 4 4 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4
4 4 4 4 1
  [81] 4 4 4 1 4 1 1 1 1 4 3 3 3 4 2 4 4 4 1 2 4 4 1 4 1 1 1 1 4 4 4 4 3 4 4
4 3 1 4 3
 [121] 1 4 1 1 4 1 4 3 1 4 3 4 1 2 1 2 1 4 1 3 4 4 2 4 4 1 4 1 4 4 4 4 3 4 4
4 4 4 1 4
 [161] 2 4 4 4 2 1 3 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 4
4 4 1 4 4
 [201] 3 4 4 1 1 4 4 4 4 3 4 1 4 4 1 4 1 1 4 4 1 1 1 1 4 4 1 1 4 3 1 4 1 4 4
4 3 4 1 1
 [241] 1 1 1 1 1 1 1 1 1 1 4 4 4 4 2 1 1 1 4 4 4 4 1 4 4 1 4 4 1 1 4 1 2 4 4
1 1 4 4 1
 [281] 1 4 1 4 1 4 1 1 4 1 4 4 4 4 4 4 4 1 1 1 4 1 4 1 4 4 4 1 2 4 4 4 4 1 1
4 3 4 4 3
 [321] 4 4 4 1 1 1 1 4 4 3 1 4 4 4 4 4 4 1 1 1 4 4 1 4 1 4 4 3 4 3 1 4 4 3 4
4 4 4 4 4
 [361] 4 2 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 4 4 1 1 4 4 1 4 4 3
2 3 4 4 4
 [401] 4 1 3 1 4 4 4 4 1 1 4 3 2 4 4 4 3 4 4 4 4 4 4 1 1 4 4 4 4 4 4 4 2 4 4
4 4 2 4 4
 [441] 1 4 1 4 1 1 4 4 4 4 4 4 1 4 4 4 4 4 2 1 4 4 4 4 4 4 4 4 2 4 4 4 4 2 4
2 3 4 1 1
 [481] 1 1 4 4 1 1 1 4 3 1 4 4 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
4 1 4 4 1
```

```
 [521] 4 4 4 1 1 4 1 4 4 4 4 1 1 3 1 1 4 4 4 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 1 1 1 2
 [561] 4 4 4 2 4 4 4 1 1 1 4 1 1 1 4 1 3 4 3 4 1 3 4 4 4 4 3 1 3 3 3 1 1 4 3
3 4 4 1 3
 [601] 1 4 2 4 1 3 4 3 1 4 4 1 1 3 4 4 1 4 3 1 1 4 4 4 1 4 4 4 2 1 1 1 1 1 1
1 1 1 1 1
 [641] 1 1 1 1 1 1 1 1 1 1 1 4 4 1 4 4 1 4 1 4 4 2 4 4 1 4 4 4 4 4 1 4 4 4 4
4 4 4 4 4
 [681] 4 4 4 1 1 1 2 4 4 4 4 4 1 4 4 1 1 4 1 4 4 1 4 4 1 1 1 4 2 4 4 3 1 4 3
4 4 4 4 4
 [721] 3 1 4 4 1 4 4 4 4 4 1 4 3 4 1 1 4 4 4 4 4 4 1 1 4 4 4 1 4 4 1 4 1 1 1
1 1 1 1 1
 [761] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 4 4 4 4 1 4 4 1 4 2 4 4 4 3 1 1 3 1
3 4 4 4 4
 [801] 1 3 1 4 3 1 4 3 4 4 4 1 4 1 1 4 4 1 4 4 1 4 4 1 4 1 1 1 4 1 4 2 4 4 4
1 4 4 4 1
 [841] 4 1 4 4 4 4 4 4 4 4 4 1 4 1 4 3 4 4 4 1 4 4 4 4 4 4 1 4 4 4 4 4 1 4 1
4 3 4 1 1
 [881] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 4 4 1 1 4 1 4 4 4 3 4 4 4
3 2 1 4 4
 [921] 4 1 1 1 1 3 4 4 3 4 4 3 4 4 4 4 4 4 4 4 3 4 4 4 4 4 1 1 1 4 4 4 4 1 4
2 4 1 4 4
 [961] 4 4 1 4 2 1 4 1 1 4 4 4 4 4 4 4 4 1 4 3 3 1 4 4 4 4 4 3 1 1 4 4 1 4 4
4 1 4 1 1
 [ reached getOption("max.print") -- omitted 22359 entries]
Levels: 1 2 3 4
> str(LargeMatrixPred)
 Factor w/ 4 levels "1","2","3","4": 4 2 4 4 3 4 4 1 3 1 ...

> summary(LargeMatrixPred)
    1     2     3     4 
10571   655  1865 10268 
```



IV. GALAXY SENTIMENT ANALYSIS

```
> galaxyDF <- galaxysmallmatrix
```

#DEPENDENT VARIABLE AS A FACTOR

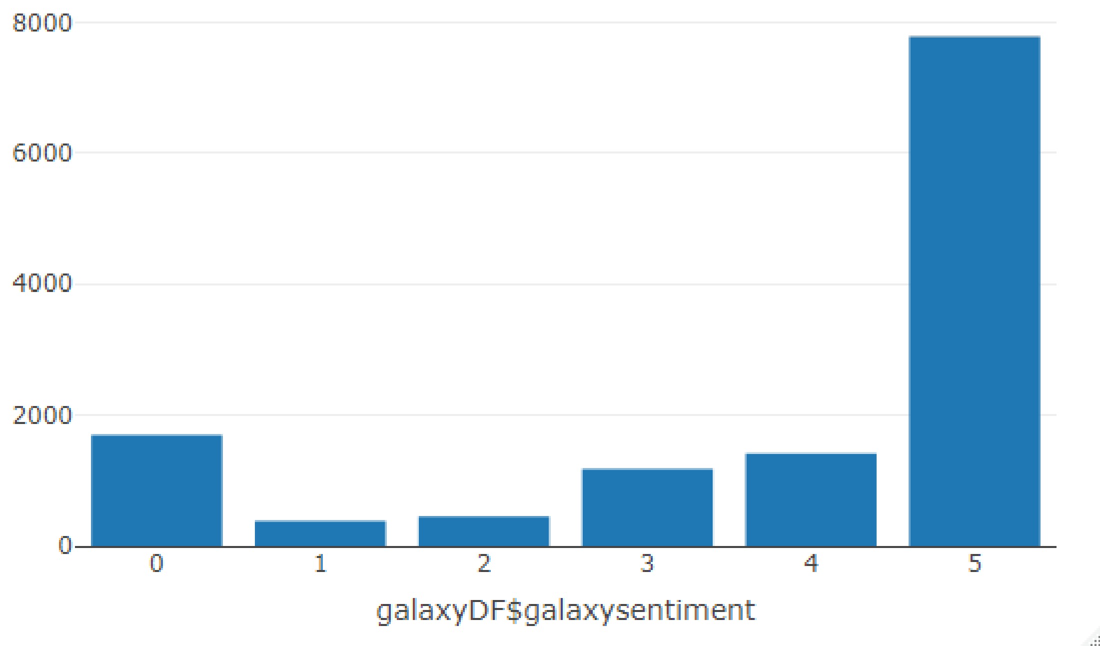
```
> galaxyDF$galaxysentiment <- as.factor(galaxyDF$galaxysentiment)
>
> str(galaxyDF)
Classes ‘spec_tbl_df’, ‘tbl_df’, ‘tbl’ and 'data.frame':     12911 obs. of  5
9 variables:
 $ iphone         : num  1 1 1 0 1 2 1 1 4 1 ...
 $ samsunggalaxy  : num  0 0 1 0 0 0 0 0 0 0 ...
 $ sonyxperia     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokialumina    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcphone       : num  0 0 0 1 0 0 0 0 0 0 ...
 $ ios            : num  0 0 0 0 0 0 0 0 0 0 ...
 $ googleandroid  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonecampos   : num  0 0 1 0 0 1 0 0 0 0 ...
 $ samsungcampos  : num  0 0 1 0 0 0 0 0 0 0 ...
 $ sonycampos     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiacampos    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htccampos      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonecamneg   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungcamneg  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonycamneg     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiacamneg    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htccamneg      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonecamunc   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungcamunc  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonycamunc     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiacamunc    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htccamunc      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonedispos   : num  0 1 0 0 0 0 2 0 0 0 ...
 $ samsungdispos  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonydispos     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiadispos    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcdispos      : num  0 0 0 1 0 0 0 0 0 0 ...
 $ iphonedisneg   : num  0 1 0 0 0 0 0 0 0 0 ...
 $ samsungdisneg  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonydisneg     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiadisneg    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcdisneg      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonedisunc   : num  0 1 0 0 0 0 0 0 0 0 ...
 $ samsungdisunc  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonydisunc     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiadisunc    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcdisunc      : num  0 0 0 1 0 0 0 0 0 0 ...
 $ iphoneperpos   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungperpos  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonyperpos     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiaperpos    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcperpos      : num  0 0 0 1 0 0 0 0 0 0 ...
```

```
 $ iphoneperneg   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungperneg  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonyperneg     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiaperneg    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcperneg      : num  0 0 0 1 0 0 0 0 0 0 ...
 $ iphoneperunc   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungperunc  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonyperunc     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiaperunc    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcperunc      : num  0 0 0 1 0 0 0 0 0 0 ...
 $ iosperpos      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ googleperpos   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iosperneg      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ googleperneg   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iosperunc      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ googleperunc   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ galaxysentiment: Factor w/ 6 levels "0","1","2","3",..: 6 4 4 1 2 1 4 6 6
6 ...
 - attr(*, "spec")=
  .. cols(
  ..   iphone = col_double(),
  ..   samsunggalaxy = col_double(),
  ..   sonyxperia = col_double(),
  ..   nokialumina = col_double(),
  ..   htcphone = col_double(),
  ..   ios = col_double(),
  ..   googleandroid = col_double(),
  ..   iphonecampos = col_double(),
  ..   samsungcampos = col_double(),
  ..   sonycampos = col_double(),
  ..   nokiacampos = col_double(),
  ..   htccampos = col_double(),
  ..   iphonecamneg = col_double(),
  ..   samsungcamneg = col_double(),
  ..   sonycamneg = col_double(),
  ..   nokiacamneg = col_double(),
  ..   htccamneg = col_double(),
  ..   iphonecamunc = col_double(),
  ..   samsungcamunc = col_double(),
  ..   sonycamunc = col_double(),
  ..   nokiacamunc = col_double(),
  ..   htccamunc = col_double(),
  ..   iphonedispos = col_double(),
  ..   samsungdispos = col_double(),
  ..   sonydispos = col_double(),
  ..   nokiadispos = col_double(),
  ..   htcdispos = col_double(),
  ..   iphonedisneg = col_double(),
  ..   samsungdisneg = col_double(),
  ..   sonydisneg = col_double(),
  ..   nokiadisneg = col_double(),
  ..   htcdisneg = col_double(),
  ..   iphonedisunc = col_double(),
  ..   samsungdisunc = col_double(),
  ..   sonydisunc = col_double(),
  ..   nokiadisunc = col_double(),
  ..   htcdisunc = col_double(),
  ..   iphoneperpos = col_double(),
  ..   samsungperpos = col_double(),
  ..   sonyperpos = col_double(),
  ..   nokiaperpos = col_double(),
  ..   htcperpos = col_double(),
  ..   iphoneperneg = col_double(),
  ..   samsungperneg = col_double(),
```

```
..    sonyperneg = col_double(),
..    nokiaperneg = col_double(),
..    htcperneg = col_double(),
..    iphoneperunc = col_double(),
..    samsungperunc = col_double(),
..    sonyperunc = col_double(),
..    nokiaperunc = col_double(),
..    htcperunc = col_double(),
..    iosperpos = col_double(),
..    googleperpos = col_double(),
..    iosperneg = col_double(),
..    googleperneg = col_double(),
..    iosperunc = col_double(),
..    googleperunc = col_double(),
..    galaxysentiment = col_double()
.. )
```

```
> library(plotly)
```

```
> plot_ly(galaxyDF, x= ~galaxyDF$galaxysentiment, type='histogram')
```



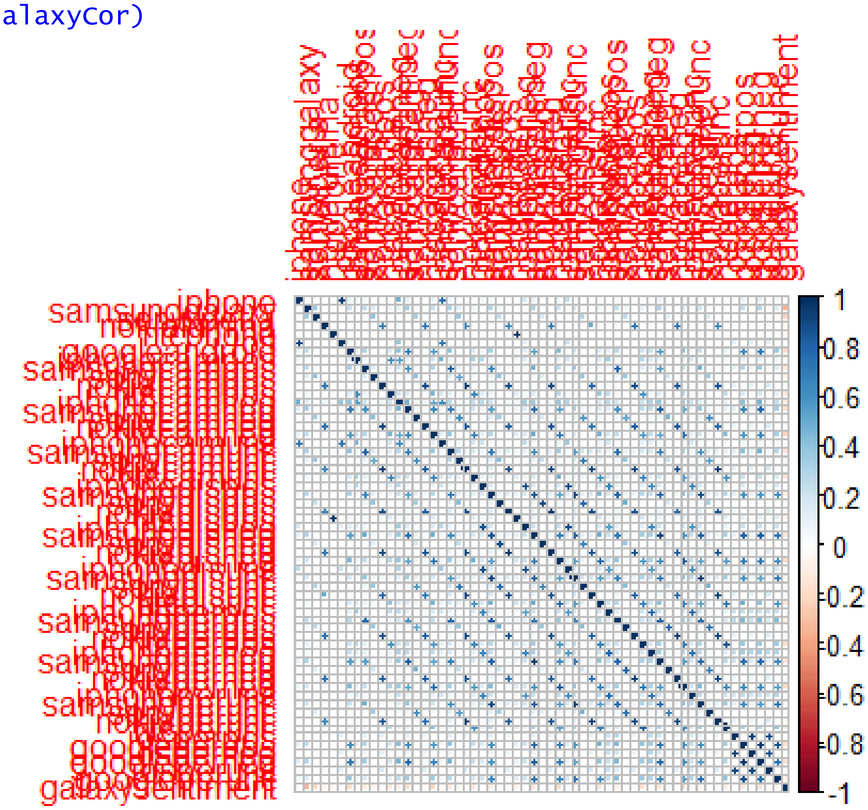
MISSING VALUES

```
> sum(is.na(galaxyDF))
[1] 0
```

==
There is no Missing Value in the Dataset.
==

CORRELATION

```
> galaxyCor <- cor(galaxyDF)
> corrplot(galaxyCor)
```



==

From the corrplot figure above, we can see that there are no such highly correlated features with the dependant variable to remove.

==

NEARZEROVARIANCE()

```
> nzvLarge <- nearZeroVar(galaxysmallmatrix, saveMetrics = TRUE)
> nzvLarge
                freqRatio percentUnique zeroVar   nzv
iphone           5.039313    0.20912400   FALSE FALSE
samsunggalaxy   14.090164    0.05421733   FALSE FALSE
sonyxperia      44.111888    0.03872667   FALSE  TRUE
nokialumina    495.500000    0.02323600   FALSE  TRUE
htcphone        11.427740    0.06970800   FALSE FALSE
ios             27.662132    0.04647200   FALSE  TRUE
googleandroid   61.248780    0.04647200   FALSE  TRUE
```

```
iphonecampos       10.526217    0.23236000   FALSE FALSE
samsungcampos      93.176471    0.08519867   FALSE  TRUE
sonycampos        347.081081    0.05421733   FALSE  TRUE
nokiacampos      1841.285714    0.08519867   FALSE  TRUE
htccampos          79.401274    0.17039734   FALSE  TRUE
iphonecamneg       19.660473    0.13167067   FALSE  TRUE
samsungcamneg      99.648438    0.06970800   FALSE  TRUE
sonycamneg       1842.428571    0.04647200   FALSE  TRUE
nokiacamneg      2148.500000    0.06196267   FALSE  TRUE
htccamneg          92.992593    0.11618000   FALSE  TRUE
iphonecamunc       16.805436    0.16265200   FALSE FALSE
samsungcamunc      73.953488    0.06970800   FALSE  TRUE
sonycamunc        585.545455    0.03872667   FALSE  TRUE
nokiacamunc      2578.800000    0.05421733   FALSE  TRUE
htccamunc          50.510040    0.12392533   FALSE  TRUE
iphonedispos        6.797333    0.24785067   FALSE FALSE
samsungdispos      96.595420    0.13167067   FALSE  TRUE
sonydispos        329.512821    0.06196267   FALSE  TRUE
nokiadispos      1431.888889    0.09294400   FALSE  TRUE
htcdispos          64.383420    0.20137867   FALSE  TRUE
iphonedisneg       10.104816    0.18588800   FALSE FALSE
samsungdisneg      98.674419    0.10843467   FALSE  TRUE
sonydisneg       2149.000000    0.06970800   FALSE  TRUE
nokiadisneg      1841.285714    0.08519867   FALSE  TRUE
htcdisneg          88.063380    0.14716134   FALSE  TRUE
iphonedisunc       11.527865    0.20912400   FALSE FALSE
samsungdisunc      74.333333    0.09294400   FALSE  TRUE
sonydisunc        757.941176    0.05421733   FALSE  TRUE
nokiadisunc      1611.625000    0.04647200   FALSE  TRUE
htcdisunc          50.757085    0.13941600   FALSE  TRUE
iphoneperpos        9.299184    0.18588800   FALSE FALSE
samsungperpos      93.748148    0.10843467   FALSE  TRUE
sonyperpos        414.903226    0.06196267   FALSE  TRUE
nokiaperpos      2147.666667    0.08519867   FALSE  TRUE
htcperpos          74.371257    0.19363334   FALSE  TRUE
iphoneperneg       11.037910    0.17039734   FALSE FALSE
samsungperneg     101.158730    0.10068933   FALSE  TRUE
sonyperneg       2149.333333    0.07745333   FALSE  TRUE
nokiaperneg      3221.750000    0.09294400   FALSE  TRUE
htcperneg          93.969925    0.15490667   FALSE  TRUE
iphoneperunc       13.034602    0.12392533   FALSE FALSE
samsungperunc      86.087838    0.09294400   FALSE  TRUE
sonyperunc       3225.000000    0.04647200   FALSE  TRUE
nokiaperunc      1841.571429    0.06970800   FALSE  TRUE
htcperunc          50.015936    0.15490667   FALSE  TRUE
iosperpos         152.626506    0.09294400   FALSE  TRUE
googleperpos       98.115385    0.06970800   FALSE  TRUE
iosperneg         141.055556    0.09294400   FALSE  TRUE
googleperneg       98.922481    0.08519867   FALSE  TRUE
iosperunc         135.234043    0.07745333   FALSE  TRUE
googleperunc       95.977444    0.07745333   FALSE  TRUE
galaxysentiment     4.593750    0.04647200   FALSE FALSE
```

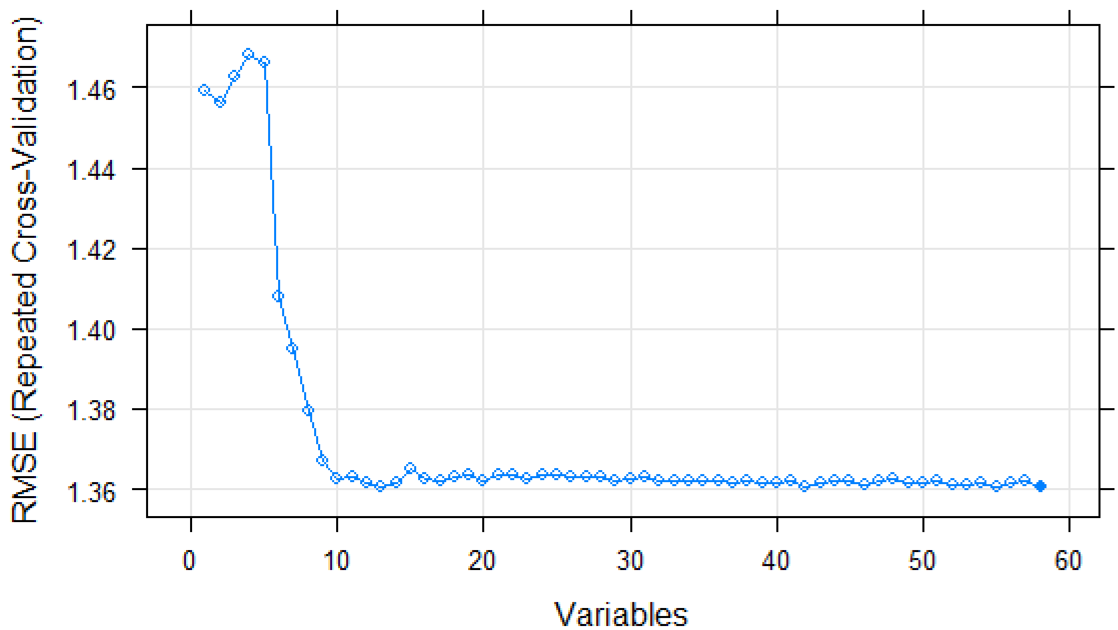
```
> nzvgalaxy <- nearZeroVar(iphoneDF, saveMetrics = FALSE)

> nzvgalaxy
 [1]  3  4  6  7  9 10 11 12 13 14 15 16 17 19 20 21 22 24 25 26 27 29 30 31
32 34 35
[28] 36 37 39 40 41 42 44 45 46 47 49 50 51 52 53 54 55 56 57 58

> galaxynzv <- galaxysmallmatrix[,-nzvgalaxy]
```

RECURSIVE FEATURE ELIMINATION

```
> galaxySample <- galaxyDF[sample(1:nrow(galaxyDF), 1000, replace=FALSE),]

> rfeResults <- rfe(galaxySample[,1:58], galaxySample$galaxysentiment, sizes=
(1:58),     rfeControl=ctrl)

> rfeResults

Recursive feature selection

Outer resampling method: Cross-Validated (10 fold, repeated 5 times)

Resampling performance over subset size:

 Variables Accuracy  Kappa AccuracySD KappaSD Selected
         1   0.6836 0.3189    0.01974 0.05656         
         2   0.6804 0.3116    0.02069 0.05518         
         3   0.6756 0.3049    0.01976 0.05384         
         4   0.7062 0.3930    0.02389 0.05839         
         5   0.7137 0.4153    0.02717 0.06541         
         6   0.7155 0.4184    0.02567 0.06242         
         7   0.7187 0.4275    0.02751 0.06564         
         8   0.7208 0.4300    0.02713 0.06536         
         9   0.7220 0.4483    0.02786 0.06408         
        10   0.7272 0.4607    0.02878 0.06524         
        11   0.7304 0.4673    0.02861 0.06443         
        12   0.7308 0.4651    0.02731 0.06262         
        13   0.7327 0.4670    0.02917 0.06891         
        14   0.7337 0.4669    0.02620 0.06363         
        15   0.7368 0.4718    0.02550 0.06212         
        16   0.7406 0.4925    0.02513 0.06048         
        17   0.7434 0.4958    0.02631 0.06151        *
        18   0.7410 0.4883    0.02510 0.06056         
        19   0.7402 0.4822    0.02565 0.06299         
        20   0.7386 0.4751    0.02388 0.06018         
        21   0.7362 0.4678    0.02462 0.06125         
        22   0.7348 0.4623    0.02508 0.06357         
        23   0.7354 0.4614    0.02505 0.06275         
        24   0.7360 0.4620    0.02633 0.06481         
        25   0.7360 0.4692    0.02502 0.06240         
        26   0.7352 0.4663    0.02404 0.06016         
        27   0.7342 0.4630    0.02429 0.06015         
        28   0.7338 0.4610    0.02491 0.06151         
        29   0.7346 0.4610    0.02648 0.06465         
        30   0.7352 0.4616    0.02498 0.06280         
        31   0.7368 0.4634    0.02577 0.06353         
        32   0.7356 0.4609    0.02605 0.06415         
        33   0.7360 0.4621    0.02549 0.06290         
        34   0.7358 0.4609    0.02670 0.06498         
        35   0.7360 0.4605    0.02552 0.06273         
        36   0.7366 0.4642    0.02585 0.06364         
        37   0.7364 0.4638    0.02615 0.06487         
        38   0.7372 0.4645    0.02564 0.06377         
        39   0.7366 0.4629    0.02531 0.06237         
        40   0.7366 0.4631    0.02645 0.06423         
        41   0.7358 0.4603    0.02571 0.06254         
        42   0.7354 0.4592    0.02664 0.06551         
        43   0.7348 0.4573    0.02655 0.06587         
        44   0.7342 0.4559    0.02696 0.06650         
        45   0.7352 0.4575    0.02696 0.06707         
        46   0.7338 0.4541    0.02671 0.06615         
        47   0.7326 0.4507    0.02724 0.06856         
        48   0.7324 0.4505    0.02706 0.06822         
        49   0.7348 0.4581    0.02635 0.06406         
        50   0.7352 0.4584    0.02503 0.06181         
```

```
 51    0.7350 0.4574    0.02638 0.06584
 52    0.7350 0.4570    0.02621 0.06560
 53    0.7332 0.4531    0.02677 0.06632
 54    0.7332 0.4523    0.02654 0.06666
 55    0.7324 0.4504    0.02743 0.06859
 56    0.7322 0.4496    0.02770 0.06936
 57    0.7306 0.4450    0.02648 0.06715
 58    0.7312 0.4463    0.02670 0.06795

The top 5 variables (out of 17):
   iphone, samsunggalaxy, googleandroid, htcphone, iphonedisunc
```

```
plot(rfeResults, type=c("g", "o"))
```




```
> galaxyRFE <- galaxyDF[,predictors(rfeResults)]

> galaxyRFE$galaxysentiment <- galaxyDF$galaxysentiment
> str(galaxyRFE)
Classes ‘spec_tbl_df’, ‘tbl_df’, ‘tbl’ and 'data.frame':     12911 obs. of  5
9 variables:
 $ iphone         : num  1 1 1 0 1 2 1 1 4 1 ...
 $ googleandroid  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonedisunc   : num  0 1 0 0 0 0 0 0 0 0 ...
 $ samsunggalaxy  : num  0 0 1 0 0 0 0 0 0 0 ...
 $ iphonedisneg   : num  0 1 0 0 0 0 0 0 0 0 ...
 $ iphoneperpos   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcphone       : num  0 0 0 1 0 0 0 0 0 0 ...
 $ iphonedispos   : num  0 1 0 0 0 0 2 0 0 0 ...
 $ htccampos      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonyxperia     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonecamneg   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonecampos   : num  0 0 1 0 0 1 0 0 0 0 ...
 $ iosperpos      : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
$ iphonecamunc   : num  0 0 0 0 0 0 0 0 0 0 ...
$ iphoneperneg   : num  0 0 0 0 0 0 0 0 0 0 ...
$ iphoneperunc   : num  0 0 0 0 0 0 0 0 0 0 ...
$ ios            : num  0 0 0 0 0 0 0 0 0 0 ...
$ htccamneg      : num  0 0 0 0 0 0 0 0 0 0 ...
$ htcdispos      : num  0 0 0 1 0 0 0 0 0 0 ...
$ htcperpos      : num  0 0 0 1 0 0 0 0 0 0 ...
$ iosperunc      : num  0 0 0 0 0 0 0 0 0 0 ...
$ iosperneg      : num  0 0 0 0 0 0 0 0 0 0 ...
$ htccamunc      : num  0 0 0 0 0 0 0 0 0 0 ...
$ htcdisunc      : num  0 0 0 1 0 0 0 0 0 0 ...
$ htcperneg      : num  0 0 0 1 0 0 0 0 0 0 ...
$ samsungperneg  : num  0 0 0 0 0 0 0 0 0 0 ...
$ htcperunc      : num  0 0 0 1 0 0 0 0 0 0 ...
$ samsungperunc  : num  0 0 0 0 0 0 0 0 0 0 ...
$ samsungdispos  : num  0 0 0 0 0 0 0 0 0 0 ...
$ sonydispos     : num  0 0 0 0 0 0 0 0 0 0 ...
$ samsungcampos  : num  0 0 1 0 0 0 0 0 0 0 ...
$ samsungperpos  : num  0 0 0 0 0 0 0 0 0 0 ...
$ samsungdisneg  : num  0 0 0 0 0 0 0 0 0 0 ...
$ googleperpos   : num  0 0 0 0 0 0 0 0 0 0 ...
$ samsungcamunc  : num  0 0 0 0 0 0 0 0 0 0 ...
$ googleperneg   : num  0 0 0 0 0 0 0 0 0 0 ...
$ htcdisneg      : num  0 0 0 0 0 0 0 0 0 0 ...
$ samsungcamneg  : num  0 0 0 0 0 0 0 0 0 0 ...
$ samsungdisunc  : num  0 0 0 0 0 0 0 0 0 0 ...
$ googleperunc   : num  0 0 0 0 0 0 0 0 0 0 ...
$ sonycampos     : num  0 0 0 0 0 0 0 0 0 0 ...
$ sonyperneg     : num  0 0 0 0 0 0 0 0 0 0 ...
$ sonyperpos     : num  0 0 0 0 0 0 0 0 0 0 ...
$ sonydisneg     : num  0 0 0 0 0 0 0 0 0 0 ...
$ sonycamneg     : num  0 0 0 0 0 0 0 0 0 0 ...
$ nokiacamneg    : num  0 0 0 0 0 0 0 0 0 0 ...
$ nokiacampos    : num  0 0 0 0 0 0 0 0 0 0 ...
$ nokiacamunc    : num  0 0 0 0 0 0 0 0 0 0 ...
$ nokiadisneg    : num  0 0 0 0 0 0 0 0 0 0 ...
$ nokiadispos    : num  0 0 0 0 0 0 0 0 0 0 ...
$ nokiadisunc    : num  0 0 0 0 0 0 0 0 0 0 ...
$ nokialumina    : num  0 0 0 0 0 0 0 0 0 0 ...
$ nokiaperneg    : num  0 0 0 0 0 0 0 0 0 0 ...
$ nokiaperpos    : num  0 0 0 0 0 0 0 0 0 0 ...
$ nokiaperunc    : num  0 0 0 0 0 0 0 0 0 0 ...
$ sonycamunc     : num  0 0 0 0 0 0 0 0 0 0 ...
$ sonydisunc     : num  0 0 0 0 0 0 0 0 0 0 ...
$ sonyperunc     : num  0 0 0 0 0 0 0 0 0 0 ...
$ galaxysentiment: num  5 3 3 0 1 0 3 5 5 5 ...
- attr(*, "spec")=List of 3
 ..$ cols   :List of 59
 .. ..$ iphone         : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ samsunggalaxy  : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ sonyxperia     : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ nokialumina    : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ htcphone       : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ ios            : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ googleandroid  : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ iphonecampos   : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
```

```
 .. ..$ samsungcampos  : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ sonycampos     : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ nokiacampos    : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ htccampos      : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ iphonecamneg   : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ samsungcamneg  : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ sonycamneg     : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ nokiacamneg    : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ htccamneg      : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ iphonecamunc   : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ samsungcamunc  : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ sonycamunc     : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ nokiacamunc    : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ htccamunc      : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ iphonedispos   : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ samsungdispos  : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ sonydispos     : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ nokiadispos    : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ htcdispos      : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ iphonedisneg   : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ samsungdisneg  : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ sonydisneg     : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ nokiadisneg    : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ htcdisneg      : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ iphonedisunc   : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ samsungdisunc  : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ sonydisunc     : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ nokiadisunc    : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ htcdisunc      : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ iphoneperpos   : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ samsungperpos  : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
 .. ..$ sonyperpos     : list()
 .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
```

```
  .. ..$ nokiaperpos    : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ htcperpos      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iphoneperneg   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ samsungperneg  : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ sonyperneg     : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ nokiaperneg    : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ htcperneg      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iphoneperunc   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ samsungperunc  : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ sonyperunc     : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ nokiaperunc    : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ htcperunc      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iosperpos      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ googleperpos   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iosperneg      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ googleperneg   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iosperunc      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ googleperunc   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ galaxysentiment: list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  ..$ default: list()
  .. ..- attr(*, "class")= chr  "collector_guess" "collector"
  ..$ skip   : num 1
  ..- attr(*, "class")= chr "col_spec"
>
> galaxyRFE$galaxysentiment <- as.factor(galaxyRFE$galaxysentiment)


> str(galaxyRFE)
Classes ‘spec_tbl_df’, ‘tbl_df’, ‘tbl’ and 'data.frame':     12911 obs. of  5
9 variables:
 $ iphone          : num  1 1 1 0 1 2 1 1 4 1 ...
 $ googleandroid   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonedisunc    : num  0 1 0 0 0 0 0 0 0 0 ...
 $ samsunggalaxy   : num  0 0 1 0 0 0 0 0 0 0 ...
 $ iphonedisneg    : num  0 1 0 0 0 0 0 0 0 0 ...
 $ iphoneperpos    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcphone        : num  0 0 0 1 0 0 0 0 0 0 ...
 $ iphonedispos    : num  0 1 0 0 0 0 2 0 0 0 ...
 $ htccampos       : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonyxperia      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonecamneg    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonecampos    : num  0 0 1 0 0 1 0 0 0 0 ...
 $ iosperpos       : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonecamunc    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphoneperneg    : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
 $ iphoneperunc   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ ios            : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htccamneg      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcdispos      : num  0 0 0 1 0 0 0 0 0 0 ...
 $ htcperpos      : num  0 0 0 1 0 0 0 0 0 0 ...
 $ iosperunc      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iosperneg      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htccamunc      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcdisunc      : num  0 0 0 1 0 0 0 0 0 0 ...
 $ htcperneg      : num  0 0 0 1 0 0 0 0 0 0 ...
 $ samsungperneg  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcperunc      : num  0 0 0 1 0 0 0 0 0 0 ...
 $ samsungperunc  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungdispos  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonydispos     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungcampos  : num  0 0 1 0 0 0 0 0 0 0 ...
 $ samsungperpos  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungdisneg  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ googleperpos   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungcamunc  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ googleperneg   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcdisneg      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungcamneg  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungdisunc  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ googleperunc   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonycampos     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonyperneg     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonyperpos     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonydisneg     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonycamneg     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiacamneg    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiacampos    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiacamunc    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiadisneg    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiadispos    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiadisunc    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokialumina    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiaperneg    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiaperpos    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiaperunc    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonycamunc     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonydisunc     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonyperunc     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ galaxysentiment: Factor w/ 6 levels "0","1","2","3",..: 6 4 4 1 2 1 4 6 6
6 ...
 - attr(*, "spec")=List of 3
  ..$ cols   :List of 59
  .. ..$ iphone         : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ samsunggalaxy  : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ sonyxperia     : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ nokialumina    : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ htcphone       : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ ios            : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ googleandroid  : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iphonecampos   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ samsungcampos  : list()
```

```
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonycampos     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiacampos    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htccampos      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphonecamneg   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungcamneg  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonycamneg     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiacamneg    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htccamneg      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphonecamunc   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungcamunc  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonycamunc     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiacamunc    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htccamunc      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphonedispos   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungdispos  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonydispos     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiadispos    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htcdispos      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphonedisneg   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungdisneg  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonydisneg     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiadisneg    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htcdisneg      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphonedisunc   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungdisunc  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonydisunc     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiadisunc    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htcdisunc      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphoneperpos   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungperpos  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonyperpos     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiaperpos    : list()
```

```
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htcperpos      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphoneperneg   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungperneg  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonyperneg     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiaperneg    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htcperneg      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphoneperunc   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungperunc  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonyperunc     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiaperunc    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htcperunc      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iosperpos      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ googleperpos   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iosperneg      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ googleperneg   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iosperunc      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ googleperunc   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ galaxysentiment: list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
..$ default: list()
.. ..- attr(*, "class")= chr  "collector_guess" "collector"
..$ skip   : num 1
..- attr(*, "class")= chr "col_spec"
```

OUT OF THE BOX MODELING GALAXY SMALL MATRIX

1) Random Forest

```
> set.seed(998)

> IntrainingGDF<- createDataPartition(galaxyDF$galaxysentiment, p=.70, list=F
ALSE)

> TrainingGDF <- galaxyDF[IntrainingGDF,]

> TestingGDF <- galaxyDF[-IntrainingGDF,]

> rfgalaxyDF <- train(galaxysentiment~., data = TrainingGDF, method = "rf", t
rControl=fitcontrol, tuneGrid=rfGrid)
> rfgalaxyDF
Random Forest

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

```
No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 8137, 8137, 8135, 8136, 8135, 8135, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  1     0.6350654  0.1260195
  2     0.7058621  0.3585410
  3     0.7157071  0.3900581
  4     0.7334057  0.4404884
  5     0.7434706  0.4666586

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 5.
```

2) C5.0

```
> CgalaxyDF<- train(galaxysentiment ~ ., data = TrainingGDF, method = "C5.0",
trcontrol=fitcontrol, tuneLength = 5)

> CgalaxyDF

C5.0

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

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 9040, 9040, 9040, 9040, 9040, 9040, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.7596882  0.5215027
  rules  FALSE   10      0.7584017  0.5170866
  rules  FALSE   20      0.7584017  0.5170866
  rules  FALSE   30      0.7584017  0.5170866
  rules  FALSE   40      0.7584017  0.5170866
  rules   TRUE    1      0.7591595  0.5208814
  rules   TRUE   10      0.7585554  0.5170130
  rules   TRUE   20      0.7585554  0.5170130
  rules   TRUE   30      0.7585554  0.5170130
  rules   TRUE   40      0.7585554  0.5170130
  tree   FALSE    1      0.7562366  0.5165461
  tree   FALSE   10      0.7580289  0.5160357
  tree   FALSE   20      0.7580289  0.5160357
  tree   FALSE   30      0.7580289  0.5160357
  tree   FALSE   40      0.7580289  0.5160357
  tree    TRUE    1      0.7561869  0.5165229
  tree    TRUE   10      0.7570776  0.5139482
  tree    TRUE   20      0.7570776  0.5139482
  tree    TRUE   30      0.7570776  0.5139482
  tree    TRUE   40      0.7570776  0.5139482

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.
```

3) SVM

```
> svmgalaxyDF <- train(galaxysentiment ~., data = TrainingGDF, method = "svmL
inear", trControl=fitcontrol, preProcess = c("center", "scale"), tuneLength =
10)
> svmgalaxyDF
Support Vector Machines with Linear Kernel

9040 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: 8137, 8138, 8135, 8136, 8135, 8135, ...
Resampling results:

  Accuracy   Kappa
  0.6973466  0.3693473

Tuning parameter 'C' was held constant at a value of 1
```

4) KKNN

```
> kknngalaxyDF <- train.kknn(galaxysentiment ~ ., data = TrainingGDF, kmax =
100, kernel = c("optimal","rectangular", "inv", "gaussian", "triangular"), sc
ale = TRUE)
>
> kknngalaxyDF

Call:
train.kknn(formula = galaxysentiment ~ ., data = TrainingGDF,     kmax = 100,
kernel = c("optimal", "rectangular", "inv", "gaussian",         "triangular")
, scale = TRUE)

Type of response variable: nominal
Minimal misclassification: 0.238385
Best kernel: inv
Best k: 16
```

POSTRESAMPLE()

```
> CpredGDF <- predict(rfgalaxyDF, TestingGDF)
> postResample(CpredGDF,TestingGDF$galaxysentiment)
 Accuracy     Kappa
0.7538104 0.4898487


> rfpredGDF <- predict(rfgalaxyDF, TestingGDF)
> postResample(rfpredGDF, TestingGDF$galaxysentiment)
 Accuracy     Kappa
0.7538104 0.4898487

> SVMpredGDF <- predict(svmgalaxyDF, TestingGDF)
> postResample(SVMpredGDF, TestingGDF$galaxysentiment)
 Accuracy     Kappa
0.7088608 0.3963808

> kknnpredGDF <- predict(kknngalaxyDF, TestingGDF)
> postResample(kknnpredGDF, TestingGDF$galaxysentiment)
 Accuracy     Kappa
0.7569104 0.5176900
```

CONFUSION MATRIX

Random Forest

```
> CMrfpredG <- confusionMatrix(rfpredGDF, TestingGDF$galaxysentiment)
> CMrfpredG
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5
         0  362    2    5    3    7   23
         1    0    0    0    0    0    0
         2    0    0   15    1    1    0
         3    2    0    1  127    2    5
         4    4    1    0    2  114    9
         5  140  111  114  219  301 2300

Overall Statistics
                                          
               Accuracy : 0.7538          
                 95% CI : (0.7399, 0.7673)
    No Information Rate : 0.6037          
    P-Value [Acc > NIR] : < 2.2e-16       
                                          
                  Kappa : 0.4898          
                                          
 Mcnemar's Test P-Value : NA              

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity           0.71260  0.00000 0.111111  0.36080  0.26824   0.9842
Specificity           0.98811  1.00000 0.999465  0.99716  0.99536   0.4231
Pos Pred Value        0.90050      NaN 0.882353  0.92701  0.87692   0.7221
Neg Pred Value        0.95791  0.97055 0.968864  0.93974  0.91687   0.9461
Prevalence            0.13123  0.02945 0.034875  0.09093  0.10979   0.6037
Detection Rate        0.09352  0.00000 0.003875  0.03281  0.02945   0.5942
Detection Prevalence  0.10385  0.00000 0.004392  0.03539  0.03358   0.8228
Balanced Accuracy     0.85035  0.50000 0.555288  0.67898  0.63180   0.7036
```

C5.0

```
> CMCpredG <- confusionMatrix(CpredGDF, TestingGDF$galaxysentiment)
> CMCpredG
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5
         0  362    2    5    3    7   23
         1    0    0    0    0    0    0
         2    0    0   15    1    1    0
         3    2    0    1  127    2    5
         4    4    1    0    2  114    9
         5  140  111  114  219  301 2300

Overall Statistics
                                          
               Accuracy : 0.7538          
                 95% CI : (0.7399, 0.7673)
    No Information Rate : 0.6037          
    P-Value [Acc > NIR] : < 2.2e-16       
```

```
                  Kappa : 0.4898

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity           0.71260  0.00000 0.111111  0.36080  0.26824   0.9842
Specificity           0.98811  1.00000 0.999465  0.99716  0.99536   0.4231
Pos Pred Value        0.90050      NaN 0.882353  0.92701  0.87692   0.7221
Neg Pred Value        0.95791  0.97055 0.968864  0.93974  0.91687   0.9461
Prevalence            0.13123  0.02945 0.034875  0.09093  0.10979   0.6037
Detection Rate        0.09352  0.00000 0.003875  0.03281  0.02945   0.5942
Detection Prevalence  0.10385  0.00000 0.004392  0.03539  0.03358   0.8228
Balanced Accuracy     0.85035  0.50000 0.555288  0.67898  0.63180   0.7036
```

SVM

```
> CMSVMpredG <- confusionMatrix(SVMpredGDF, TestingGDF$galaxysentiment)
> CMSVMpredG
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5
         0  302    6    3   13   24   70
         1    0    0    0    0    0    1
         2    4    1    4    0    0    1
         3   44    0   15  111    3    5
         4    1    1    0    2   70    3
         5  157  106  113  226  328 2257

Overall Statistics

               Accuracy : 0.7089
                 95% CI : (0.6943, 0.7231)
    No Information Rate : 0.6037
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3964

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0  Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity           0.59449 0.0000000 0.029630  0.31534  0.16471   0.9658
Specificity           0.96551 0.9997338 0.998394  0.98096  0.99797   0.3937
Pos Pred Value        0.72249 0.0000000 0.400000  0.62360  0.90909   0.7082
Neg Pred Value        0.94034 0.9705426 0.966071  0.93474  0.90643   0.8830
Prevalence            0.13123 0.0294498 0.034875  0.09093  0.10979   0.6037
Detection Rate        0.07802 0.0000000 0.001033  0.02867  0.01808   0.5831
Detection Prevalence  0.10798 0.0002583 0.002583  0.04598  0.01989   0.8233
Balanced Accuracy     0.78000 0.4998669 0.514012  0.64815  0.58134   0.6798
```

KKNN

```
> CMkknnpredG <- confusionMatrix(kknnpredGDF, TestingGDF$galaxysentiment)
> CMkknnpredG
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5
         0  355    2    3    8   14   40
```

```
         1    0    1    1    0    0    0
         2    2    0   15    1    2    4
         3    5    2    3  211    6   26
         4    2    2    3    4  109   28
         5  144  107  110  128  294 2239

Overall Statistics

               Accuracy : 0.7569
                 95% CI : (0.7431, 0.7704)
    No Information Rate : 0.6037
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5177

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 0  Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity           0.69882 0.0087719 0.111111  0.59943  0.25647   0.9581
Specificity           0.98008 0.9997338 0.997591  0.98806  0.98868   0.4896
Pos Pred Value        0.84123 0.5000000 0.625000  0.83399  0.73649   0.7409
Neg Pred Value        0.95564 0.9707935 0.968807  0.96103  0.91512   0.8846
Prevalence            0.13123 0.0294498 0.034875  0.09093  0.10979   0.6037
Detection Rate        0.09171 0.0002583 0.003875  0.05451  0.02816   0.5784
Detection Prevalence  0.10902 0.0005167 0.006200  0.06536  0.03823   0.7807
Balanced Accuracy     0.83945 0.5042529 0.554351  0.79375  0.62258   0.7238
```

==

*The classifier that had the best performance regarding this modeling was **KKNN** with an accuracy value of **0.7569** and kappa value of **0.5177** except the evaluation of its confusion metrics wasn't showing good results in the Positive Pred Value which make it an inaccurate classification.*

*For those reasons I have decided to go for either **C5.0** or **Random Forest**, both came across with very similar accuracy and kappa and better evaluation in each of their respective Confusion Matrix.*

==

PREDICTIONS OUT OF THE BOX GALAXY MODELING USING THE BEST CLASSIFIERS

C5.0

```
> CpredGDF <- predict(CgalaxyDF, TestingGDF)
> summary(CpredGDF)
   0    1    2    3    4    5 
 393    0   19  245  139 3075 
```

Random Forest

```
> rfpredGDF <- predict(rfgalaxyDF, TestingGDF)
> summary(rfpredGDF)
   0    1    2    3    4    5 
 402    0   17  137  130 3185 
```

ENGINEERING THE DEPENDANT VARIABLE "galaxysentiment"

```
> library(dplyr)
```

```
> galaxyRC$galaxysentiment <- recode(galaxyRC$galaxysentiment, '0' = 1, '1' =
1, '2' = 2, '3' = 3, '4' = 4, '5' = 4)


> galaxyRC$galaxysentiment <- as.factor(galaxyRC$galaxysentiment)

> str(galaxyRC)
Classes ‘spec_tbl_df’, ‘tbl_df’, ‘tbl’ and 'data.frame':     12911 obs. of  5
9 variables:
 $ iphone        : num  1 1 1 0 1 2 1 1 4 1 ...
 $ samsunggalaxy : num  0 0 1 0 0 0 0 0 0 0 ...
 $ sonyxperia    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokialumina   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcphone      : num  0 0 0 1 0 0 0 0 0 0 ...
 $ ios           : num  0 0 0 0 0 0 0 0 0 0 ...
 $ googleandroid : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonecampos  : num  0 0 1 0 0 1 0 0 0 0 ...
 $ samsungcampos : num  0 0 1 0 0 0 0 0 0 0 ...
 $ sonycampos    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiacampos   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htccampos     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonecamneg  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungcamneg : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonycamneg    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiacamneg   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htccamneg     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonecamunc  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungcamunc : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonycamunc    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiacamunc   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htccamunc     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonedispos  : num  0 1 0 0 0 0 2 0 0 0 ...
 $ samsungdispos : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonydispos    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiadispos   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcdispos     : num  0 0 0 1 0 0 0 0 0 0 ...
 $ iphonedisneg  : num  0 1 0 0 0 0 0 0 0 0 ...
 $ samsungdisneg : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonydisneg    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiadisneg   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcdisneg     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iphonedisunc  : num  0 1 0 0 0 0 0 0 0 0 ...
 $ samsungdisunc : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonydisunc    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiadisunc   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcdisunc     : num  0 0 0 1 0 0 0 0 0 0 ...
 $ iphoneperpos  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungperpos : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonyperpos    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiaperpos   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcperpos     : num  0 0 0 1 0 0 0 0 0 0 ...
 $ iphoneperneg  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungperneg : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonyperneg    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiaperneg   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcperneg     : num  0 0 0 1 0 0 0 0 0 0 ...
 $ iphoneperunc  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ samsungperunc : num  0 0 0 0 0 0 0 0 0 0 ...
 $ sonyperunc    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ nokiaperunc   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ htcperunc     : num  0 0 0 1 0 0 0 0 0 0 ...
 $ iosperpos     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ googleperpos  : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
 $ iosperneg      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ googleperneg   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ iosperunc      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ googleperunc   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ galaxysentiment: Factor w/ 4 levels "1","2","3","4": 4 3 3 1 1 1 3 4 4 4 .
..
 - attr(*, "spec")=List of 3
  ..$ cols   :List of 59
  .. ..$ iphone          : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ samsunggalaxy   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ sonyxperia      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ nokialumina     : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ htcphone        : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ ios             : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ googleandroid   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iphonecampos    : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ samsungcampos   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ sonycampos      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ nokiacampos     : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ htccampos       : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iphonecamneg    : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ samsungcamneg   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ sonycamneg      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ nokiacamneg     : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ htccamneg       : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iphonecamunc    : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ samsungcamunc   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ sonycamunc      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ nokiacamunc     : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ htccamunc       : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iphonedispos    : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ samsungdispos   : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ sonydispos      : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ nokiadispos     : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ htcdispos       : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
  .. ..$ iphonedisneg    : list()
  .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
```

```
.. ..$ samsungdisneg  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonydisneg     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiadisneg    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htcdisneg      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphonedisunc   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungdisunc  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonydisunc     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiadisunc    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htcdisunc      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphoneperpos   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungperpos  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonyperpos     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiaperpos    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htcperpos      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphoneperneg   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungperneg  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonyperneg     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiaperneg    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htcperneg      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iphoneperunc   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ samsungperunc  : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ sonyperunc     : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ nokiaperunc    : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ htcperunc      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iosperpos      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ googleperpos   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iosperneg      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ googleperneg   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ iosperunc      : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ googleperunc   : list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
.. ..$ galaxysentiment: list()
.. .. ..- attr(*, "class")= chr  "collector_double" "collector"
..$ default: list()
.. ..- attr(*, "class")= chr  "collector_guess" "collector"
```

```
  ..$ skip   : num 1
  ..- attr(*, "class")= chr "col_spec"
```

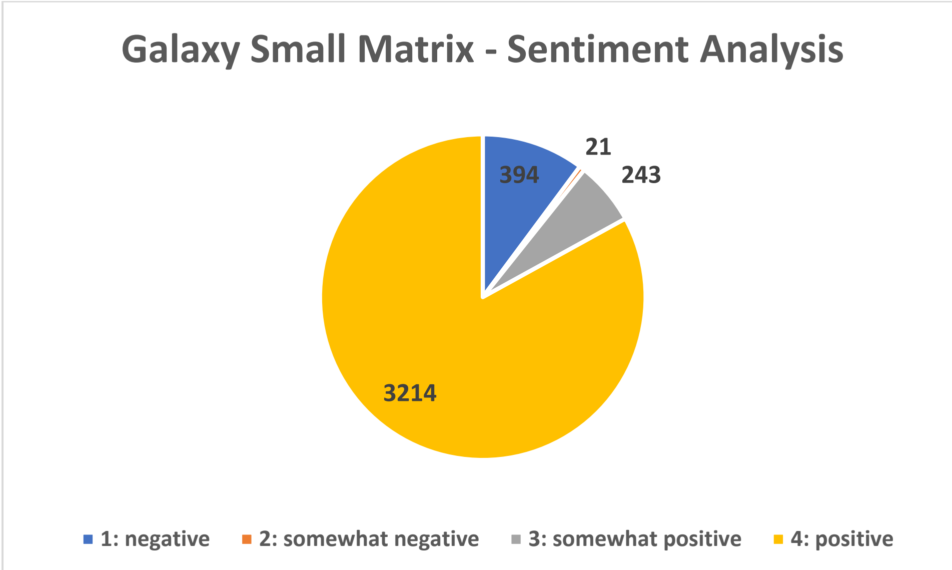
Random Forest

```
> IntrainingGRC<- createDataPartition(galaxyRC$galaxysentiment, p=.70, list=F
ALSE)

> TrainingGRC <- galaxyRC[IntrainingGRC,]
> TestingGRC <- galaxyRC[-IntrainingGRC,]

> rfgalaxyRC <- train(galaxysentiment~., data = TrainingGRC, method = "rf", t
rControl=fitcontrol, tuneGrid=rfGrid)


> rfpredGRC <- predict(rfgalaxyRC, TestingGRC)
>
> summary(rfpredGRC)
   1    2    3    4
 388   20  104 3360
> postResample(rfpredGRC, TestingGRC$galaxysentiment)
 Accuracy     Kappa
0.8228306 0.5109232
> CMrfpredGRC <- confusionMatrix(rfpredGRC, TestingGRC$galaxysentiment)
> CMrfpredGRC
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3    4
         1  351    0    3   34
         2    0   19    0    1
         3    2    1   95    6
         4  270  115  254 2721

Overall Statistics

               Accuracy : 0.8228
                 95% CI : (0.8104, 0.8347)
    No Information Rate : 0.7133
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5109

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity           0.56340 0.140741  0.26989   0.9852
Specificity           0.98861 0.999732  0.99744   0.4243
Pos Pred Value        0.90464 0.950000  0.91346   0.8098
Neg Pred Value        0.92193 0.969886  0.93179   0.9199
Prevalence            0.16090 0.034866  0.09091   0.7133
Detection Rate        0.09065 0.004907  0.02454   0.7027
Detection Prevalence  0.10021 0.005165  0.02686   0.8678
Balanced Accuracy     0.77601 0.570237  0.63366   0.7047
```

C5.0

```
> CgalaxyGRC<- train(galaxysentiment ~ ., data = TrainingGRC, method = "C5.0"
, trcontrol=fitcontrol, tuneLength = 5)
```

```
> CpredGRC <- predict(CgalaxyGRC, TestingGRC)
> summary(CpredGRC)
   1    2    3    4 
 394   21  243 3214 
```




```
> postResample(CpredGRC, TestingGRC$galaxysentiment)
 Accuracy     Kappa 
0.8378099 0.5794131 
> CMCpredGRC <- confusionMatrix(CpredGRC, TestingGRC$galaxysentiment)
> CMCpredGRC
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3    4
         1  347    0    7   40
         2    1   19    0    1
         3    5    1  197   40
         4  270  115  148 2681

Overall Statistics
                                          
               Accuracy : 0.8378          
                 95% CI : (0.8258, 0.8493)
    No Information Rate : 0.7133          
    P-Value [Acc > NIR] : < 2.2e-16       
                                          
                  Kappa : 0.5794          
                                          
 Mcnemar's Test P-Value : < 2.2e-16       

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity           0.55698 0.140741  0.55966   0.9707
Specificity           0.98553 0.999465  0.98693   0.5198
Pos Pred Value        0.88071 0.904762  0.81070   0.8342
Neg Pred Value        0.92064 0.969878  0.95729   0.8769
Prevalence            0.16090 0.034866  0.09091   0.7133
Detection Rate        0.08962 0.004907  0.05088   0.6924
Detection Prevalence  0.10176 0.005424  0.06276   0.8301
Balanced Accuracy     0.77126 0.570103  0.77330   0.7452
```

SVM

```
> svmgalaxyRC <- train(galaxysentiment ~., data = TrainingGRC, method = "svmL
inear", trControl=fitcontrol, preProcess = c("center", "scale"), tuneLength =
10)

> svmpredGRC <- predict(svmgalaxyRC, TestingGRC)
> summary(svmpredGRC)
   1    2    3    4 
 359    8  165 3340 
> postResample(svmpredGRC, TestingGRC$galaxysentiment)
 Accuracy     Kappa 
0.7869318 0.4175572 
> CMsvmpredGRC <- confusionMatrix(svmpredGRC, TestingGRC$galaxysentiment)
> CMsvmpredGRC
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3    4
         1  275    2   12   70
         2    1    4    0    3
         3   44   16   92   13
         4  303  113  248 2676

Overall Statistics
                                          
               Accuracy : 0.7869          
                 95% CI : (0.7737, 0.7997)
    No Information Rate : 0.7133          
    P-Value [Acc > NIR] : < 2.2e-16       
                                          
                  Kappa : 0.4176          
                                          
 Mcnemar's Test P-Value : < 2.2e-16       

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity           0.44141 0.029630  0.26136   0.9689
Specificity           0.97415 0.998930  0.97926   0.4018
Pos Pred Value        0.76602 0.500000  0.55758   0.8012
Neg Pred Value        0.90094 0.966097  0.92986   0.8383
Prevalence            0.16090 0.034866  0.09091   0.7133
Detection Rate        0.07102 0.001033  0.02376   0.6911
Detection Prevalence  0.09272 0.002066  0.04261   0.8626
Balanced Accuracy     0.70778 0.514280  0.62031   0.6853
```

KKNN

```
> library(kknn)

> kknngalaxyRC <- train.kknn(galaxysentiment ~ ., data = TrainingGRC, kmax =
100, kernel = c("optimal","rectangular", "inv", "gaussian", "triangular"), sc
ale = TRUE)


> kknngalaxyRC <- train.kknn(galaxysentiment ~ ., data = TrainingGRC, kmax =
100, kernel = c("optimal","rectangular", "inv", "gaussian", "triangular"), sc
ale = TRUE)
>
```

```
> kknnpredGRC <- predict(kknngalaxyRC, TestingGRC)
>
> summary(kknnpredGRC)
   1    2    3    4
 423   24  247 3178
> postResample(kknnpredGRC, TestingGRC$galaxysentiment)
 Accuracy     Kappa
0.8318698 0.5699272
> CMkknnpredGRC <- confusionMatrix(kknnpredGRC, TestingGRC$galaxysentiment)

> CMkknnpredGRC
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3    4
         1  345    2   14   62
         2    1   18    0    5
         3    8    2  200   37
         4  269  113  138 2658

Overall Statistics

               Accuracy : 0.8319
                 95% CI : (0.8197, 0.8435)
    No Information Rate : 0.7133
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5699

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity            0.5538 0.133333  0.56818   0.9623
Specificity            0.9760 0.998394  0.98665   0.5315
Pos Pred Value         0.8156 0.750000  0.80972   0.8364
Neg Pred Value         0.9194 0.969595  0.95807   0.8501
Prevalence             0.1609 0.034866  0.09091   0.7133
Detection Rate         0.0891 0.004649  0.05165   0.6865
Detection Prevalence   0.1092 0.006198  0.06379   0.8208
Balanced Accuracy      0.7649 0.565864  0.77741   0.7469
```

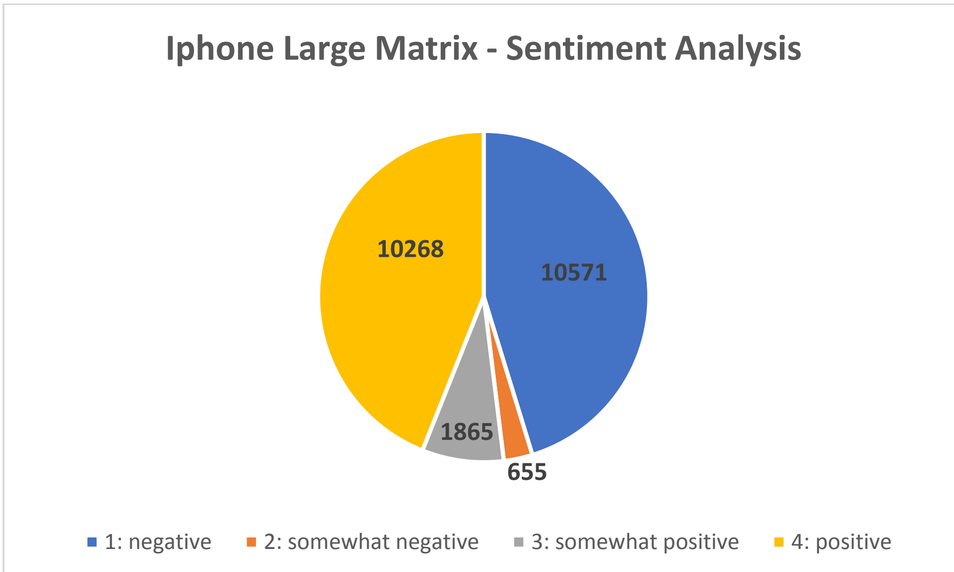
===

The classifier that had the best performance regarding the galaxy small matrix modeling using the engineering dependant variable was ***C5.0*** *with an accuracy value of 0.8378099 and kappa value of 0.5794131 with a better performance shown on the confusion matrix performance evaluation.*

===

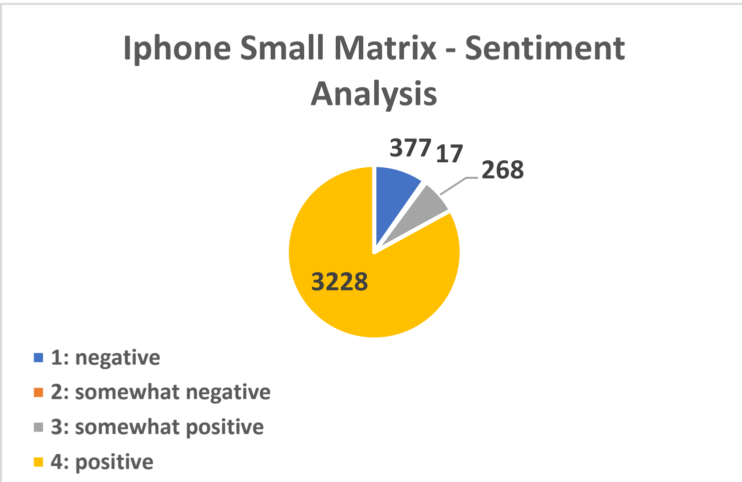
V. CONCLUSION

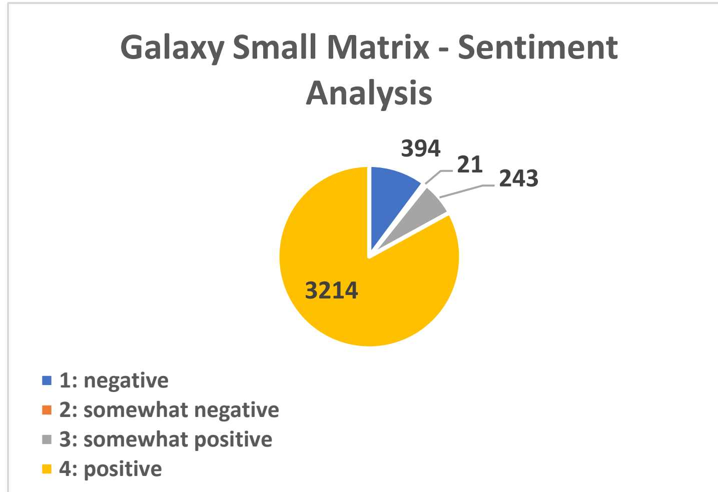
iPhone is a device that has more lovers or haters. We can see it with the sentiment distribution in the figure below showing the results of our Prediction modeling for the "iPhoneLargeMatrix":



In my sentiment analysis, I have tried to determine if they were a relationship between the variables in my datasets using the correlation Matrix, automated the features selection with the rfe() function and also used the principal component analysis.

For this subject, the engineering of the dependent variable was so far the most effective process to get better results by using the algorithm **C5.0** for both datasets of the iPhone with an a*ccuracy of **84%** and Kappa of **51%*** and Galaxy with an *Accuracy of **84%** and Kappa of **61%**,* the figures below show the outcome of this analysis:





VI. METHODOLOGY

R PROGRAMMING LANGUAGE

R is a programming language and free software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis.

The reasons why we chose R for our Data analysis is because of its series of steps; programming, transforming, discovering, modeling and communicate the results:

- **Program**: R is a clear and accessible programming tool.
- **Transform**: R is made up of a collection of libraries designed specifically for data science
- **Discover**: Investigate the data, refine your hypothesis and analyze them
- **Model**: R provides a wide array of tools to capture the right model for your data
- **Communicate**: Integrate codes, graphs, and outputs to a report with R Markdown or build Shiny apps to share with the world.

CLASSIFICATION MODELING

A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. Outcomes are labels that can be applied to a dataset.

There are a number of classification models. Classification models included in our sentiment analysis were Random Forest, C5.0, Support Vector Machine (SVM) and Weighted k-Nearest Neighbors (KKNN).

These algorithms consist of a target / outcome variable (or dependent variable which in our analysis were the iphonesentiment and galaxysentiment) which is to be predicted from a given set of predictors (independent variables like htcphone, ios, googleandroid, iphonecampos, …).

Using these set of variables, we have generated a function that map inputs to desired outputs. the training process continues until the model achieves a desired level of accuracy on the training data that lead to a better skilled modeling (in our case the best modeling was using the C5.0 algorith for both iPhone and galaxy datasets).

VII. APPENDIX (Lessons Learned)

This task had me enjoying every step as I get more familiar with the R programming language and all data processes and classifiers, this was an opportunity that gathered all the knowledge that I developed in the previous courses in one task.

I have been able to follow every step through the task and learn how to apply new processes on different datasets for comparison.

Here are some of the processes that I have learned and enjoyed working with:

Confusion Matrix: As the models through the course had a poor performance, the use of the confusion matrix allowed me to learn more about other metrics such us sensitivity, Specificity, … in order to select the best algorithm.

Engineering the dependant variable: using the recode() function was a game changing in my data distribution as I had to drop the levels of my dependant variable from 6 to 4 which made a big difference in the results of my modeling.

Principal Component Analysis: Finding the principal component was a little challenging at first, but once diving into the resources, the comprehension of the process gets much easier, unfortunately this analysis didn't help to uplift the results of our modeling.

Recursive Feature Elimination: a great form of automated feature selection using function rfe() with random forest.

In general, the task was a time consuming but a great asset to reuse all methodologies, processes and knowledge learned through the previous courses using the caret package in R programming language.