**Learner Report for Sentiment Analysis**

### Overall Impression

In this classification task, I first used library() to recall a series of previously installed packages like caret, readr, tidyr, plotly, dplyr, corrplot etc. Then I installed and applied a new package of “parallel processing” which in turn allocated 4 out of 8 cores from my local computer to process three datasets. I have 12973 \* 59 size of vector from iPhone small matrix, 12911 \* 59 size of vector from Galaxy small matrix and 22461 observations which I collected from Common Crawl for sentiment count analysis. We need to preprocess these data before feeding into six different models. Preprocessing is a vital step that involves transforming raw data into a understanable format. In this task, I created 6 different datasets for each iPhone and Samsung Galaxy. After testing on various models, I compared the Accuracy and Kappa values and Confidence Interval and P-Values for each model, I chose the feature engineering recode function on random forest model for my final prediction.

### Preprocessing

#### Step 1 – Removing collinearity?

We have 59 features in each small matrix, a lot of features are highly correlated with each other. Shall I remove these attributes? The answer was No! Because highly correlated attribute does not bother in the classification problem. However, for the sake of my experiment, I still filtered out some of these correlated features and gave names for the remaining features as: iphoneCOR and samsungCOR.

#### Step 2 – Removing the Near Zero Variance

When exploring the entire datasets, I found that there are a lot of data have the same value, i.e. 0. These are the zero or near Zero Variance data which are less informative and could be taken away. Here I applied nearZeroVar() from caret package to create an index of near zero variance features. The index will allow us to quickly remove these nearZeroVar() features from each original datasets. Then I gave two names as iphoneNZV and samsungNZV.

#### Step 3 – Recursive Feature Elimination

RFE is a form of automated feature selection. Caret’s [rfe() function](https://topepo.github.io/caret/recursive-feature-elimination.html) with random forest will try every combination of feature subsets and return a final list of recommended features. RFE does not use the outcome so it must be removed from the data set before implementation and then added back in before modeling. At this step I generated two new datasets namely: iphoneRFE and samsungRFE.

### Models Using Your Feature Selection Data Sets

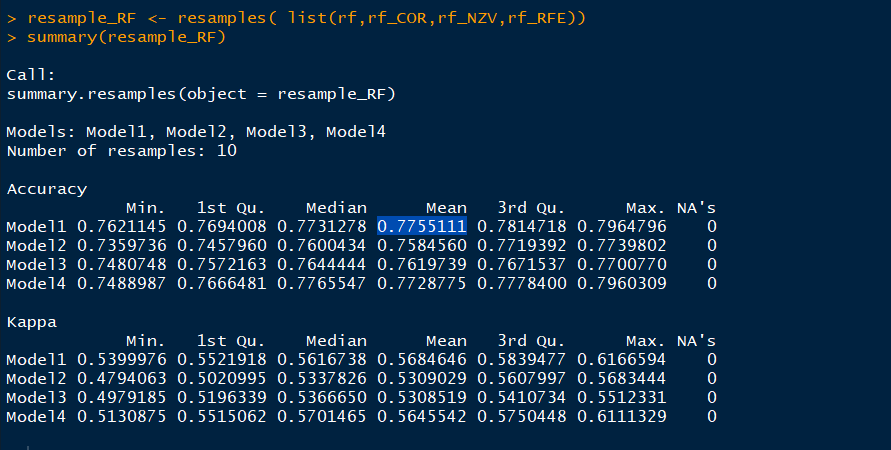
#### Out of the Box Model development

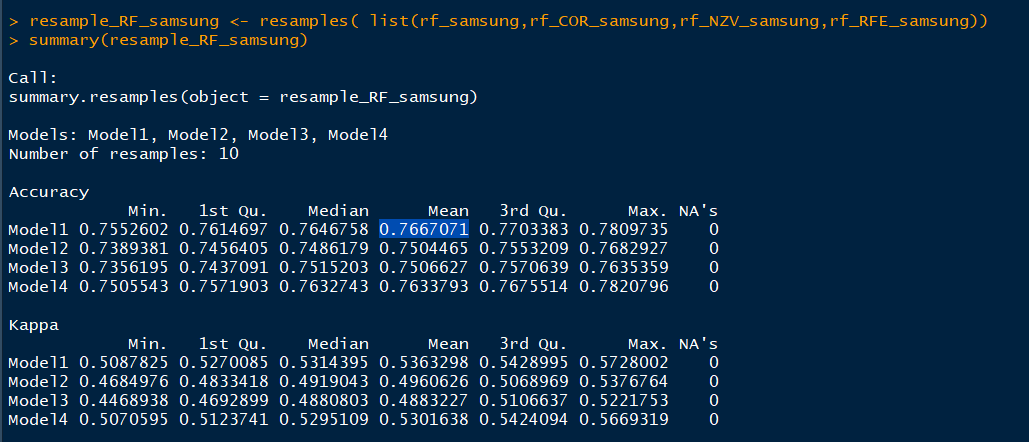
After preprocessing the datasets, it is time to apply various models to these existing datasets. The first two experiments were to apply 5 carets models, namely C5.0, Random Forest, KKNN, SVM, SVM e1071 to the original iphoneDF and samsungDF that contain all features to gain “out of the box” accuracy and Kappa. The range of Accuracy value was between 0.33-0.77 and the range of Kappa value was between 0.17-0.57. The best algorithm for the original data frames is random forest model with Accuracy: 0.78 and Kappa: 0.57 on iphoneDF and Accuracy: 0.77 and Kappa: 0.54 on samsungDF.

#### Other models/datasets combination trails

Besides the out of box model on original DFs, I also tried to run random forest model on my feature selection data sets (iphoneCOR, samsungCOR, iphoneNZV, samsungNZV, iphoneRFE, and samsungRFE).

Below screenshots show random forest model works best in combination of the original out of box dataset (iphoneDF and samsungDF)

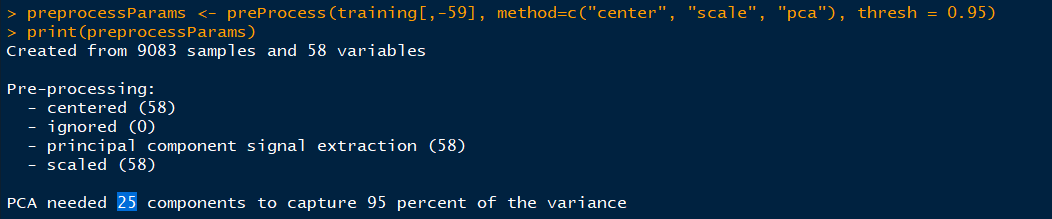


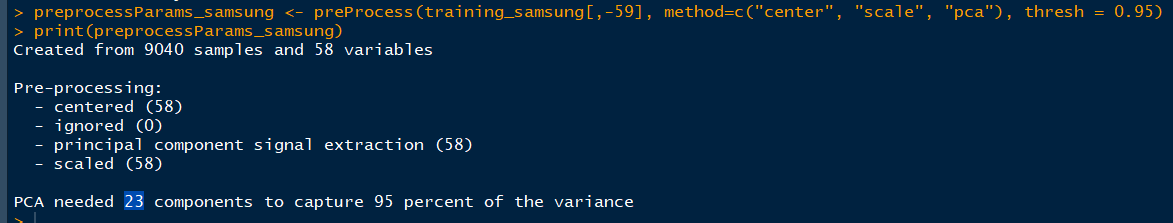


### The classifier you selected and the features (attributes) you used to train the classifier

#### Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a form of feature engineering that removes all features and replaces them with mathematical representations of their variance. The caret preprocess() function take "pca" as an argument. Setting the threshold states the amount of variance you want to PCA to capture in the model.





Above two screenshots show 25 and 23 components can be captured with 95% of the variance for iphone and Samsung respectively.

Below two screenshots show the Accuracy and Kappa value with postResample() function on the iphoneDF and samsungDF.

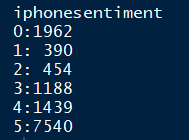
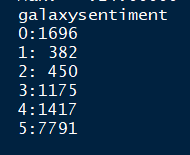




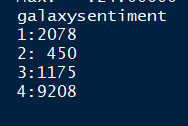
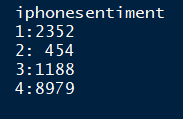
#### Feature Engineering the dependent variable

Feature Engineering is the process of using [domain knowledge](https://en.wikipedia.org/wiki/Domain_knowledge) of the data to create [features](https://en.wikipedia.org/wiki/Feature_(machine_learning)) that make [machine learning](https://en.wikipedia.org/wiki/Machine_learning) algorithms work. Initially, we have six sentiment categories for both iPhone and Galaxy derived from the original datasets ranging from 0 very negative to 5 very positive. After a series of trials and errors on various algorithms, we found that the independent variable's factor levels had very poor Sensitivity and Balanced Accuracy. Therefore, we decided to combine some of the redundant levels and reduced them to 4 levels ranging from 1-4 (negative to positive).

summary(iphoneDF) summary(samsungDF)

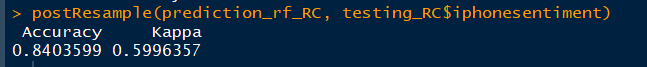
 

summary(iphoneRC) summary(samsungRC)

Here we used the dplyr package recode() function can help us with this. I found that both Accuracy and Kappa values improved dramatically after applying the ultimate recode function, simply, to reduce the factor levels from 6 to 4, which eventually lead to our best result of accuracy: 0.84 and Kappa: 0.60 for iPhone small matrix and accuracy: 0.84 and Kappa: 0.59 for Samsung Galaxy small matrix.

**This is by far the best algorithm and dataset combination! It was to apply feature engineering recode function to random forest model!**

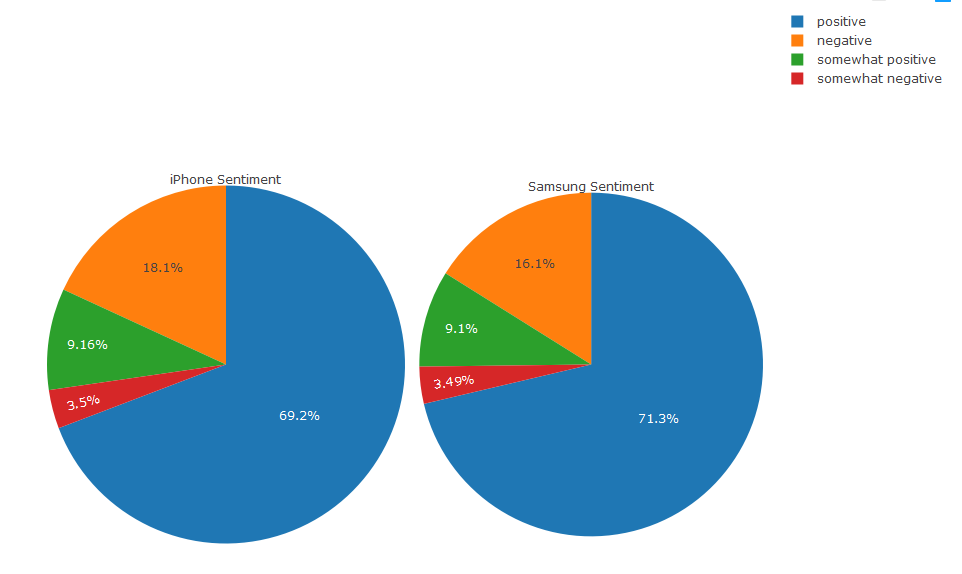




### What worked well. What didn’t work. What was difficult.

Overall, the plan of attack provides step by step detailed explanation. Therefore, I did not find any difficulty until I started to create the side by side pie chart by plotly. I referred some online articles from additional resources and tried about a couple of hours and finally I made it. The tricky part was to set the value. We need to use summary(iphoneRC) to get last column of iphonesentiment count and put the values in below values vector. The syntax is like:

pieData\_iphoneRC <- data.frame(COM = c("negative", "somewhat negative", "somewhat positive", "positive"), values = c( 2352, 454, 1188, 8979 ))



### How the process to execute similar projects should be changed for the future.

In the future, I would recommend applying feature engineering recode function to random forest model first when dealing with the sentiment analysis, as it is the best winning model and dataset combination.

In addition, we do not need to spend time to eliminate collinearity for any given classification problem. Because collinearity does not both the classification question. This has been proven by applying various models on iphone\_COR and Samsung\_COR datasets, as neither can surpass the testing results than the original out of box predictions.

The accuracy and Kappa values for KKNN or KNN model are kind of low (0.3309455 and 0.1626048 respectively) Therefore, we can save our time by not implementing this model in the future for sentiment analysis.