Smartphone Sentiment Analysis: Lessons Learned Report

January 23rd, 2020

Alert! Analytics

Author: Esteban Villalobos Gómez

Cenfotec: XTOL

Lessons learned from Smartphone Sentiment Analysis

Feature selection

For both iPhone and Galaxy, four datasets were created, each using a different method of feature selection, which were:

1. Use all available features
2. Remove highly correlated features (COR)
3. Remove Near Zero Variance (NZV) features
4. Use Recursive Feature Elimination (RFE)

For both phones, we trained four different models per algorithm, each one using one of the different feature-selected datasets, and for each algorithm, the model with better accuracy and kappa metrics was selected, leaving 4 classifiers in the end, the best one per algorithm (KNN, C5.0, Random Forest and Support Vector Machines).

For the iPhone, the selected models use the following features :

1. Random Forest: RFE
2. C5.0: RFE
3. SVM: RFE
4. KNN: Use all features.

For the Galaxy, the models were trained using the features selected as follows:

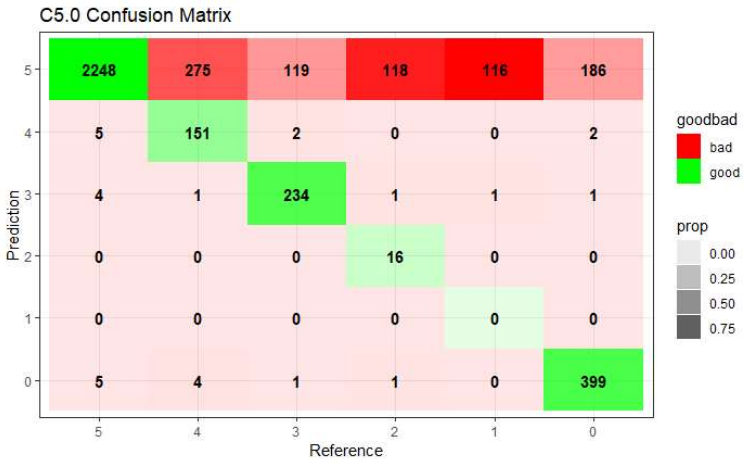
1. Random Forest: Use all features.
2. C5.0: Use all features.
3. SVM: NZV
4. KNN: COR

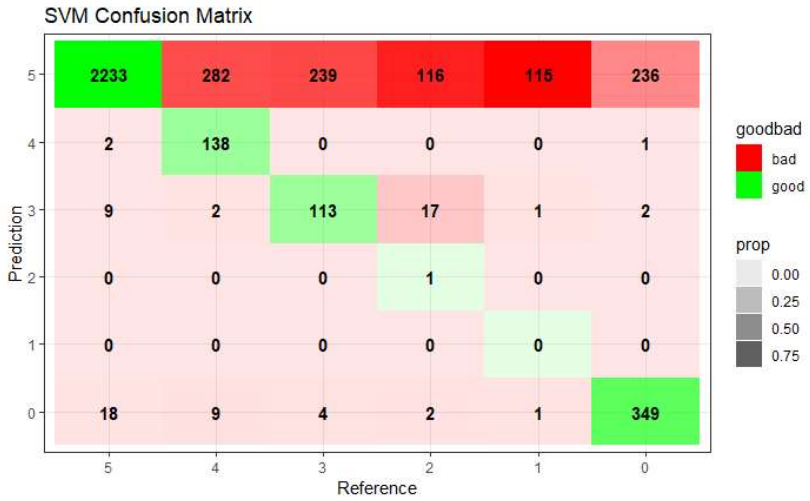
Classifier selection

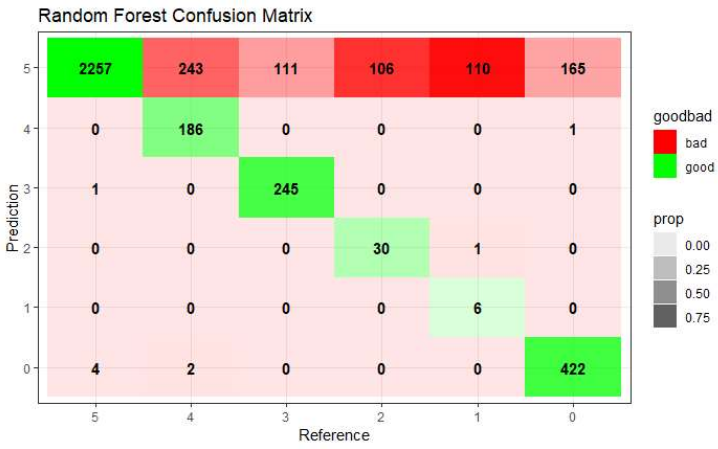
iPhone

When comparing the performance of models, I selected the accuracy and the kappa as my main drivers, and then validated with the confusion matrix that the selection was good. Using this rationale, the **Random Forest performed better with the testing dataset**, see the following **comparative performance chart**:

And by verifying the confusion matrices for all models, it was detected that they mostly performed similarly, hence the selection was based on model Accuracy and Balanced overall accuracy.

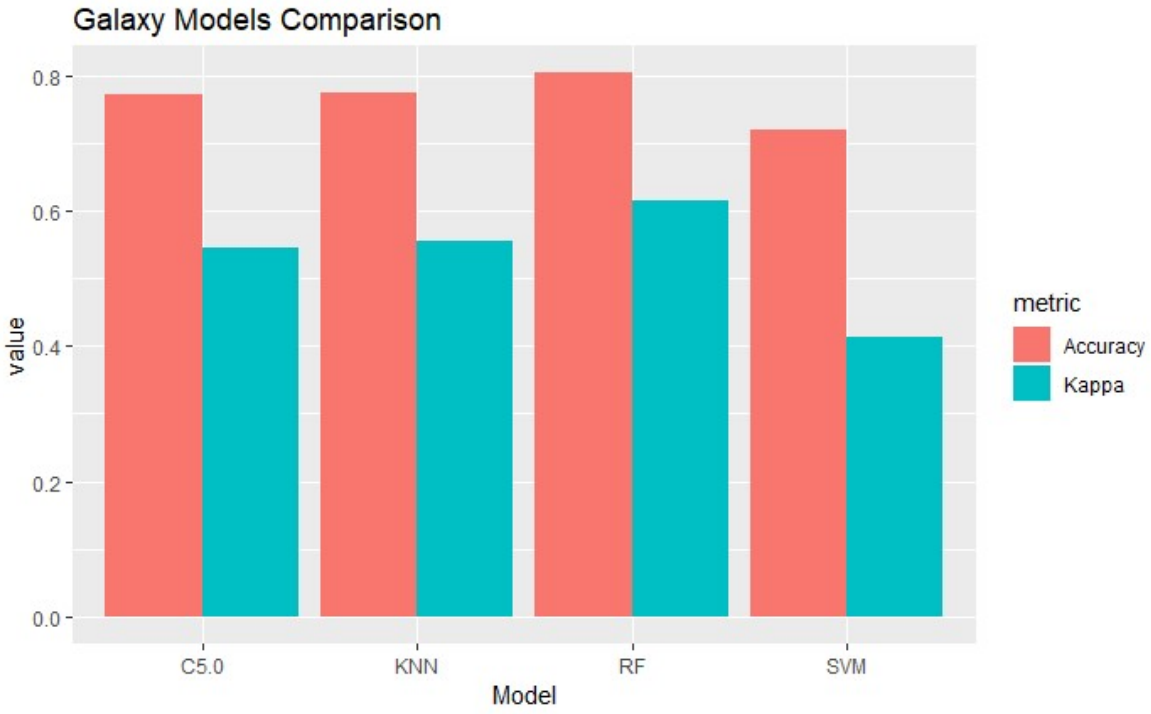






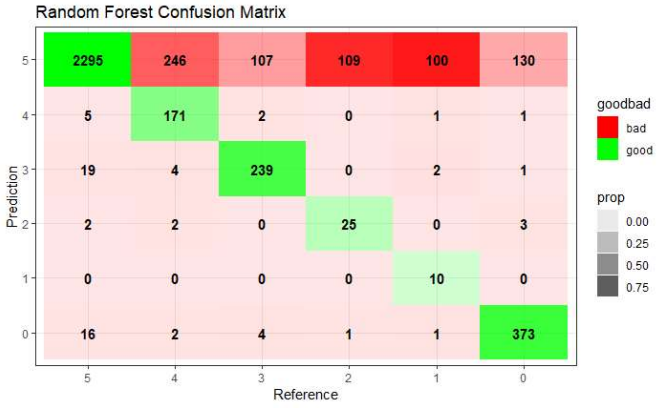
Galaxy

All models performed similarly for the Galaxy, see the following **comparative performance chart:**



Again, the **random forest** was selected using the same rationale used for the iPhone model selection.

The Galaxy models' confusion matrices were the following:

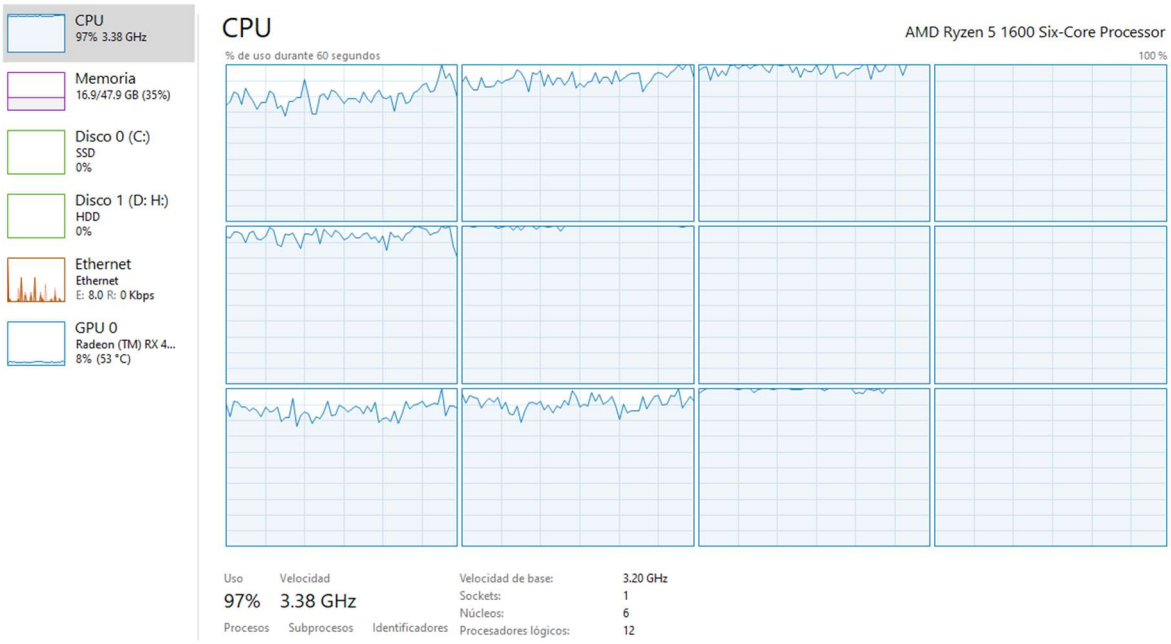


Recommendations (what worked well, what didn't work or was difficult)

Use parallelism where possible

Taking advantage of multiple CPU cores improved greatly resources usage, thus reducing training time.

The following diagram shows the usage of 12 cores while training a Random Forest algorithm for the iPhone Sentiment matrix:



Using a system with 12 cores made the training time took about 10 minutes, while using a system with just 4 cores, caused the same training to take 90 minutes.

Take advantage of online documentation

R has thousands of online resources and examples, which make it easy to accomplish the most common tasks on data exploration, cleanup and training.

Non-Caret packages require additional configuration

Using models not defined in the CARET package, cause some troubles, for example it was difficult to get the predictions from the kknn model, while using CARET is quite simple.

Justify well before using third party packages.

Confusion matrix reveals models training trends

It was interesting seen how the confusion matrix revealed how the models were failing to predict some data, balancing it to a specific class, for instance the class “Very Positive” on the iPhone training data set.

Consider reducing the number of classes in your prediction

The less classes you are trying to predict, the bigger the confidence interval you will have, which would improve accuracy, at the expense of reducing specificity. In my case, I decided to be as specific as possible with the analysis.

Use “post resamples” to get performance metrics from the testing dataset

It is better to assess model performance using the validation or testing dataset to get the accuracy and kappa metrics. If you use the model’s metrics from the training set, you will be getting a somehow biased metric, since the end goal is to see how the model behaves with unseen data.

Future projects recommendation

Sentiment analysis like this, is somehow simplistic, and it doesn’t seem very accurate, many of the counts are taken out of words without context, which reduces credibility on this analysis. Other methods using more advanced NLP toolkits should be used in the futures.

Also manually labeling the data could have biased the models, given that most of the results turned out to be negative for both smartphones.

In the future, try to reduce the number of categories (in this case there was a time constraint that prevented me for trying that).