With this assignment we expanded out from working just in our local production environment of personal computers to one that involved the virtual space of the World Wide Web. For this assignment we got the ball rolling by starting the process of creating the datasets that we would use throughout the project in Amazon Web Service an online asset that provides the ability to process, organize, and store large sets of data to be used for our client in determining the superior smart phone between iPhone and Galaxy and to create a predictive model, using the R programming language. To better answer questions and ensure that I provide insight into successes, failures, and what was learned I will include the list here of what was requested to be addressed and follow it to the best of my ability:

* For both iPhone and Galaxy:
  + The classifier you selected and the features (attributes) you used to train the classifier.
  + Your rationale for selecting the classifier that you did.
  + The features you eliminated from the data matrix and your rationale for doing so.
  + Comparative performance of the classifiers you tried (you can explain in text or with a chart).
* What worked well. What didn’t work. What was difficult.
* How the process to execute similar projects should be changed for the future.

For this project I found that the kNN (k-Nearest Neighbor) classifier worked the best for me for both iPhone and Galaxy predictions. I set up the classifier much the same way as I have before with the trainControl function using the “repeatedcv” method, number of folds set to ten, and repeats of cross validation set to ten as well – pretty standard settings for trainControl. The actual kNN function was very much customary code as well (see Example 1) with defining attribute being iphoneSentiment, which established the feeling reviewers had for iPhone smart phones.

iphoneknn=train(iphoneSentiment ~., data = iphone4ktraining, method="knn",trControl=ctrl2,preProcess=c("center","scale"),tuneLength=5)

Example 1

The reason for iphoneSentiment being chosen was that it fit the requirements for what the client most wanted to learn about from this work. If the client could predict what consumer’s thought iPhones and Galaxy phones it would improve their sales potentials at the various Blackwell stores. The preprocessing function, preProcess, was set to the default from the previous assignment of “center” and “scale” and tuneLength was the same. After doing some research online and testing some different settings for the kNN train function I did not find any reason to change it and so it was left with the same features as before from a previous assignment.

kNN algorithm was chosen in part because of the ease of implementing the code. As I mentioned before the programming I actually used for this task matched almost perfectly with the coding used in a previous assignment. The main difference being that some of the variables were changed in order to work with a different problem and training set of data. Another benefit of using kNN was that unlike the other classifiers that were tested ( Random Forest, C5.0, and SVM) kNN did not having any hang ups or high demands, such as taking a long time to process like Random Forest or extra steps for validity testing as with SVM (support vector machine).

The datasets I used for creating the training and testing sets of data for iPhone and Galaxy analysis ended up not needing to be changed. I came to this conclusion when reviewing the data at the beginning of this assignment to see if there were any issues that may be found in a dataset that is new or I have not seen it before – duplicate values, empty instances, or noise in the records. After seeing if there were any abnormalities in Excel by applying filters to the data to scan for blank cells and creating some pivot tables to check the variety of data I had I found that there were no discernable issues. Later on when I was at the phase of validating classifiers the results I got back for the accuracy and Kappa scores for kNN were in the goldilocks zone of being not too high or too low. Normally in my experience if there were any issues that I had missed in the data they would cause issues in the prediction results.

To better understand the results I got from creating ten different classifiers across two phones, iPhone and Galaxy, I made a chart detailing the accuracy and Kappa scores achieved by each of them (see Example 2).

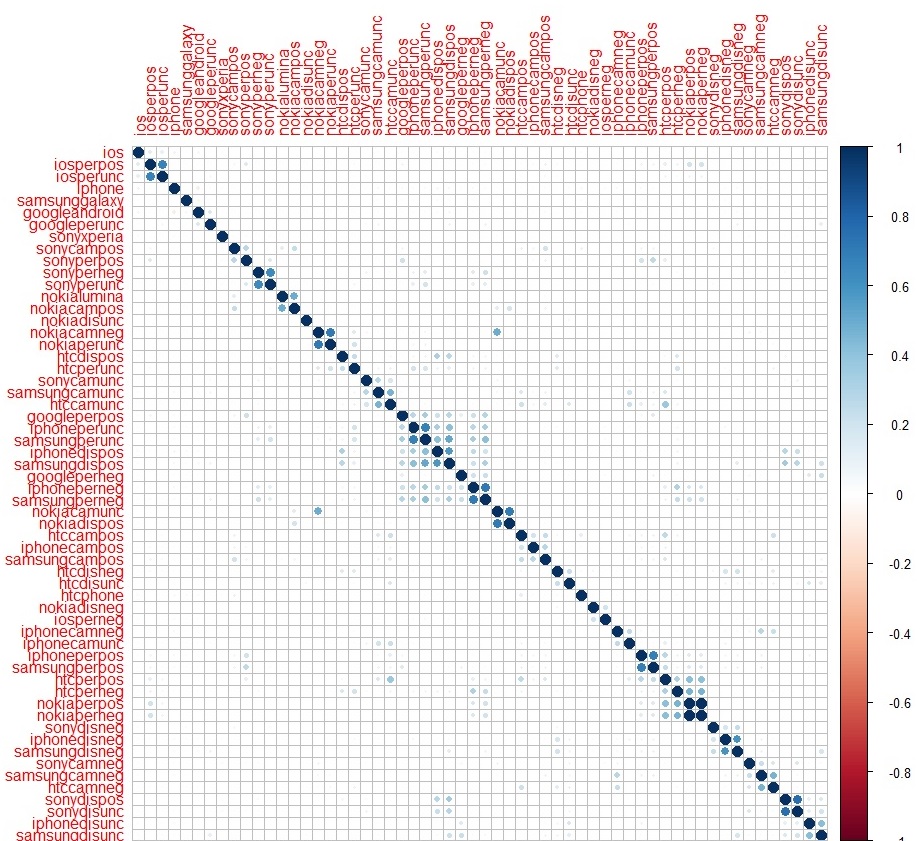
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Classifier Accuracy and Kappa Scores | | | |
|  | **iPhone** | | **Galaxy** | |
|  | **Accuracy** | **Kappa** | **Accuracy** | **Kappa** |
| **kNN** | 0.9303775 | 0.860492 | 0.956692 | 0.8612259 |
| **Random Forest (200 Trees)** | 0.9881703 | 0.97682178 | 0.9867665 | 0.95825636 |
| **Random Forest (500 Trees)** | 0.9880897 | 0.97662893 | 0.9869474 | 0.95885683 |
| **C5.0** | 0.9849372 | 0.9703272 | 0.984127 | 0.9492388 |
| **SVM** | 0.66359833 | -0.01706865 | 0.819548872 | 0.003342084 |

Example 2

* **kNN** – Following previous method of creating the code for kNN provided the best performance results of the four classifier’s chosen to test. In that the accuracy is high, but not so high that the classifier code or training and test set must be changed. The Kappa score is also great because it fits that range of being 0.8 or higher, meaning it hits a high level of agreement with the data, which we want to see in predictive models. kNN coding, as always, is straightforward to implement and has no heavy burden on computers.
* **Random Forest** – Required the most time of the four classifiers to run with each instance taking upwards of 30 minutes to complete. I had issues of RStudio freezing when trying to process the Random Forest code because of its sheer size, thus limiting the number of trees that could be used as part of the classifier’s code. Even though the training and test sets were subsets of the full matrixes of iPhone or Galaxy phone data. In order for this classifier to work it may require a more slimmed down test and training set with attributes removed to eliminate some burden.
* **C5.0** – An interesting classifier that produces decision trees for data analysis. I found that much like the Random Forest classifier the accuracy and Kappa scores were too high for both iPhone and Galaxy phone results. The decision trees themselves were interesting to look at from an analytical standpoint, but the trees produced were too detailed and hard to follow with the iPhone decision tree having 43 branches and the Galaxy 50. In order for this classifier to be functional for the project there would need to be testing for attribute removal or optimizing the code in order to improve the results, which again was not an issue for kNN being well performing out the gate.
* **SVM** – The odd duck of the four classifiers. The accuracy and Kappa score results I got were not encouraging as they were extremely low and would fail to deliver anywhere near satisfactory prediction results. After doing some research online to see how the SVM could be changed to get better results I did not see any reason to explore SVM further as I had already got such promising results from kNN.

From looking back at everything that we accomplished for this assignment I can say the part that worked best for me without causing any substantial issues was the programming work. For me I found that funny as early on with programming in R I found it to be huge issue to try and get anything to work and taking absurd amounts of time to complete, but in this work that we did it was pretty direct with everything from creating the classifiers to making the prediction function were very straightforward and caused no issues at all. And while not all of the classifiers worked as well as I would have liked them to I have learned that not every classifier is meant for every job.

The one thing that really did not work for me was creating the heat maps that were a recommendation and not a requirement early in the assignment plan. I tried a number of different methods and changes to my code to try and make them work, but nothing productive ever came out of it. And this was after researching heat maps online and visiting my trusty reference sites Quora and Stackexchange. It was over a week ago that I first tried making heat maps from the iPhone and Galaxy datasets, but nothing seems to work for me for some unknown reason. The one good thing that came from this work of trying to visualize the data was that the correlation plotting function, corrplot, worked well (see Example 3) at communicating the relationship the attributes had for each other.



Example 3

The one thing I found difficult was the production of the iPhone and Galaxy datasets through Amazon Web Services. Oddly it was not the coding that I had an issue with as the instructions were direct enough and just giving myself enough time to work through the code to ensure every line made sense at every step worked out to make sure there were no issues. The problems for me arose when trying to have any of the files that were needed to be processed in AWS. Whenever I ran a cluster the process would fail with error message “Terminated with errors” leading me to learn how to read the log files to see the detailed error message and figure out what went wrong. This lead me to finding out that when cluster processes fail that every facet of the process has to be reviewed and redone to ensure effectiveness. Thus I had to again download and install the mapper and reducer files, check the makeup of the cluster settings, and test the whole process over again until it worked. Eventually I found out that the one thing that was causing the issue, the cluster hardware settings, was not being changed and foolishly I kept using the same one for some time, which wasted a few days that could have been better spent finishing actual work for the assignment! Once I changed the master hardware to use c4.large instead of the recommended m3.large the job of having the files processed worked and every assignment involving AWS worked well from there. So the issue turned out that the default cluster hardware I was trying to use was not available for whatever reason. And so I realized I must use a reference on Amazon hardware availability in order to save time going forward.

When thinking over every task we completed for this assignment I cannot think of anything that should be changed to help improve how well work would turn out in the future. The difficult parts that I wrote about before, such as finding the right accessible server from Amazon was to me just part of the learning experience and I cannot think of changing that for any reason – we needed to think about the problems we were facing, testing to see what did and did not work, and then implementing the best option. So taking everything we worked on into consideration I would not change a thing as it all helped us to become better at what we do.