Ind Study: Machine Learning **(Due: December 10, 2020)**

Project #2

Instructor: Bill Michael *Name:* Alex Allenspach

Problem

(a) Report the **accuracy**, **number of misclassified**, **recall**, and **running time** for the three algorithms.

Table 1: Performance for Three Algorithms

Model	Accuracy	Number of Misclassified	Recall	Run Time
Naive Bayes	72.46%	27.54%	73.90%	248.06sec
Neural Networks	46.65%	53.35%	78.33%	221.13sec
Support Vector Machine	74.89%	25.11%	74.88%	476.02sec

(b) What is the **feature size** you used for the Neural Networks and the SVM algorithm?

Table 2: Feature Size

Model	Feature Size
Naive Bayes	6,082
Neural Networks	440
Support Vector Machine	440

(c) What is the **name of the library** you used to build for the Neural Networks and the SVM classifications?

Table 3: Library Name for Classifications

Model	Library
Neural Networks	nnet
Support Vector Machine	e1071

(d) Report the **accuracy**, **number of misclassified**, **recall**, and **running time** when the dictionary (Bag of Words) size is **altered** for the Naive Bayes Algorithm.

Table 4: Performance of Naive Bayes with Different Bag of Words Size

Dictionary Size	Accuracy	Number of Misclassified	Recall	Run Time
3,997	71.50%	28.50%	74.07%	245.28sec
2,134	73.41%	26.59%	73.83%	210.03sec
1,317	73.19%	26.81%	73.76%	200.98sec
568	73.49%	26.51%	73.80%	197.43sec

(e) Comment on the performance of (d) above and explain the reason for the increase or decrease in accuracy?

There is an increase in accuracy with less features being used (varies a little around 73%). I also did not strictly cut off my matrix size and instead reduced my features by making the data frames more sparse. Adding in uninformative features by trying to have a big dictionary will not increase the testing accuracy. Choosing informative features for the dictionary will lead to the best results and is also quicker at classifying, but too little features may not help either.

(f) Reduce the feature size of neural Networks and the SVM algorithm and report the **accuracy**, **number of misclassified**, **recall**, and **running time**

Table 5: Performance with Reduced Feature Size

Model	Dictionary	Accuracy	Miss-classification	Recall	Run Time
Neural Networks	213	59.95%	40.05%	76.46%	190.75sec
Support Vector Machine	213	74.22%	25.78%	74.17%	379.7sec

Comments About Project

I have code where I go into the directory and all subfolders and pull the data and do some pre-cleaning by getting rid of the headers and signatures and then split the data into the train and testing (60/40). However, everytime I tried to pass this data into the function I would get an encoding "UTF-8" error and I was not able to come up with a solution. In Python you would just have to specify the encoding and this error would not occur but this did not work in R. So I had to save the data as a .csv and read in the file using the function "fread" which takes care of the UTF-8 error. From there I just saved the data into a train.csv and test.csv so when I call any of the classification models no errors from the file encoding would happen. This is why I use the file instead of the directory path when calling functions. When I had too many features for my neural network or svm model I would get the following error: "too many (12167) weights" or "Error: cannot allocate vector of size 2.5Gb". This is why I had to make it very sparse. The run time is calculated from the very beginning of each function call instead of starting at the model classification. The cleaning of the text is identical in all functions so their run time is equivalent therefore I decided to start the clock at the start.