For the first portion of the project, we decided to include Naïve Bayes and kNN within one test file to make it easier on runtime within the command prompt. The command line arguments, including algorithm type, k number (if applicable), and the path to the folder holding the train and test data, were all provided via args[] within the test class ‘classify’. In response to the data received, the algorithm of choice is able to run accordingly, provided that the parameters have been specified correctly to it via constructor.

The second task, the experimental portion, was completed through the use of the ScikitLearn libraries, in which we experimented with Support Vector Machines (SVM) [1] and Random Forest Classifiers (RFC) [2]. The entirety of this section was contained within one file, where one could choose what algorithm they wanted to employ on the data, as well as the source to where the data was coming from.

As far as the efficiency of these algorithms go, we found SVM to be the slowest, due to the fact that larger datasets typically have a negative impact on how quickly the classification tasks can be performed. The fastest algorithm, Naïve Bayes, was largely owed to it being much more adept to larger bodies of data, as opposed to kNN, which is quick if and only if the dataset is minute in scale.

|  |  |
| --- | --- |
| *Algorithm* | *Accuracy on the Test Set* |
| *Naïve Bayes* | \_\_\_\_ % |
| *k-NN* | k=1 \_\_\_\_ %  k=3 \_\_\_\_ %  k=5 \_\_\_\_ %  k=19 \_\_\_\_ % |
| *Support Vector Machines* | 98% |
| *Random Forest Classifiers* | \_\_\_\_ % |

Works Cited

[1] KDNuggets,

URL: <https://www.kdnuggets.com/2017/03/email-spam-filtering-an-implementation-with-python-and-scikit-learn.html>

[2] Medium,

URL: <https://medium.com/machine-learning-101/chapter-5-random-forest-classifier-56dc7425c3e1>

<https://github.com/kinejohnsrud/naive-bayesian-spam-filter>