**DATA622: Homework1: Essay**

by Glen Davis

For Homework 1, I sought out labeled text datasets because I wanted to try two different supervised machine learning approaches to a multiclass classification problem. The small dataset I chose featured social media messages from Twitter, Facebook, and Instagram that were labeled by sentiment (positive, negative, or neutral), while the large dataset I chose featured Twitter messages that were labeled by emotion (sadness, joy, love, anger, fear, or surprise).

I initially wanted to compare the sentiment/emotion classification performance of a Multinomial Naïve Bayes Classifier with either a K Nearest Neighbors Model or a Random Forest Model using only the text features in both datasets. However, the large dataset contained over 400,000 observations, which forced me to make a decision that limited the algorithms I was able to use. Tokenizing text data into its component words for analysis can result in a very large number of binary categorical features even with a small dataset because every word present in the data becomes its own column, and every observation takes a value of 1 for every word column that is present in its text and a value of 0 for every word column that is not. So it quickly became clear that I would not be able to represent the data in the large dataset using a typical dataframe without encountering memory issues. There were simply too many unique words even after implementing a word frequency cut-off to eliminate those that only occurred once in the large dataset. However, the fact that most observations would have values of 0 for most words proved very helpful in finding a solution. I was able to turn the word features into sparse matrices, and I did this for both datasets for the sake of model comparison even though it wasn’t strictly necessary for the small dataset. Thankfully, Multinomial Naïve Bayes Classifiers are able to handle classification problems when the input features are represented by sparse matrices just fine, but K Nearest Neighbors models and Random Forest models are not. That eliminated two of the algorithms I was considering, so I picked Extreme Gradient Boosting (XGBoost) as my second algorithm instead.

The biggest pros of the Multinomial Naïve Bayes Classifier are that it is computationally inexpensive and not prone to overfitting, and its biggest con is that it tends to require a lot of training data in order to perform well. The biggest pro of the XGBoost Model is that even before tuning, its predictive power is often superior to other models, and its biggest cons are that it is computationally expensive and prone to overfitting. I expected the Multinomial Naïve Bayes Classifier to underperform on the small dataset, but it surprised me by beating the XGBoost Model there, and while the XGBoost Model did beat the Multinomial Naïve Bayes Classifier on the large dataset, the difference in performance was smaller than I expected. They both made much more accurate predictions on the large dataset than they did on the small dataset, and their predictive accuracy was within a couple percentage points of each other. Those extra couple percentage points of predictive accuracy might be very valuable in certain business contexts, justifying the XGBoost Model’s high computation cost.

In conclusion, both models extrapolated well to the test data when they had large amounts of training data. Neither suffered from overfitting, which is generally a concern with any tree models, so I was glad to see the XGBoost Model did not suffer there. Anytime I have a multiclass classification problem and a very large dataset in the future, I would consider both of these algorithms again. With the same problem and a smaller dataset, I probably wouldn’t have to represent the data using a sparse matrix, so I would have the advantage of being able to consider more algorithms.