Machine Learning – Bayesian Classification

Alex Kirwan - R00132481

This report is a concise evaluation of my findings while implementing a Multinomial bayesian classification model to distinguish between positive and negative movie reviews from the IMDB movie set.

Both training and testing data sets were split into positive and negative classes. On the first iteration of the testing the model a 78.25 % accuracy was recorded. Datasets were then run through a regex function removing any special characters, white spaces and converting word to lowercase. Both negative and positive class sets were sliced to ensure both datasets were of equal length. This first step of preprocessing the data set left us with a clean set with no missing features. Running the classification model with this dataset gave a accuracy of 81.05%. Removing word tokens that are found in the standard english stop word list was then implemented, this left us with a data set with a relevant features. This simple preprocessing technique added a significant boost to the accuracy of just over one percent leaving the model with a 82.9% accuracy.

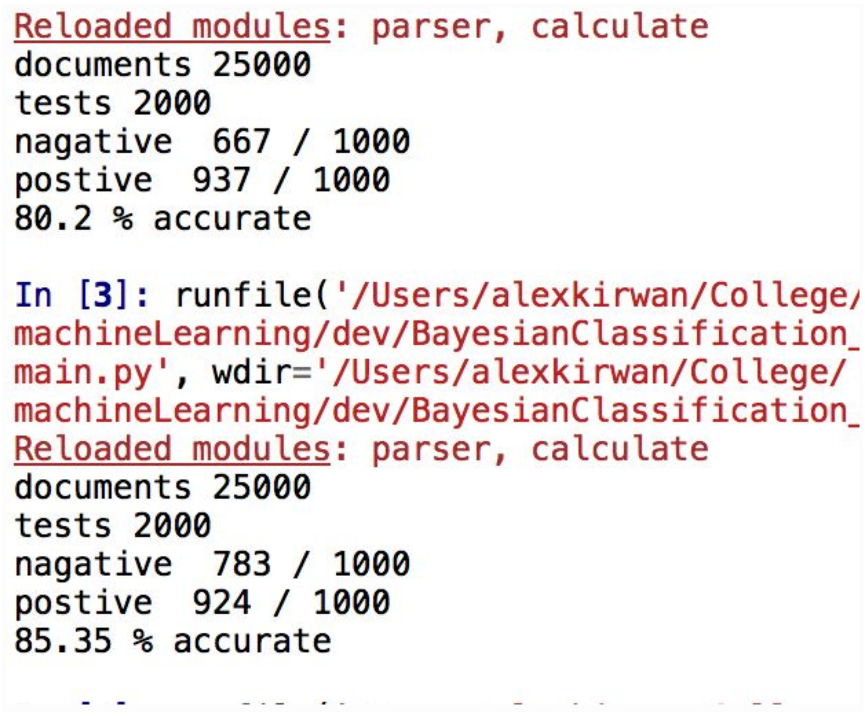
After further investigation into improving the accuracy of the classification model. The Natural language toolkit(NLTK) was then imported to utilize its extensive selection of prepossessing functions, implementing the he word_tokenize function replacing

"open(os.path.join(absPath, d, f), "r", encoding="utf8").read()"

Added

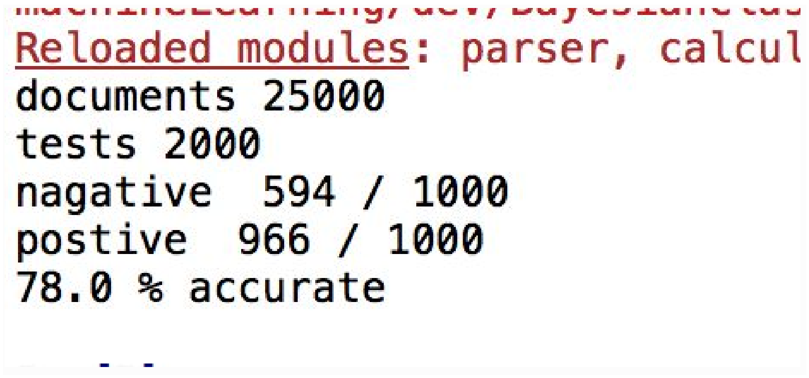
Using both lemmatizer and porter stemmer algorithm decreased my overall accuracy to 81.68% and 81.75%. This could be due to the details being removed from the features having an important role in the context of the feature.

The next pre processing strategy that was implemented was implementing a n-gram strategy where a token can be taken as a sequence of items. This can give context to a token/feature given a totally new perspective on the meaning. If you take into account a string like "This movie was not good" in a uni-gram model like we already have had implemented the tokens "not" and "good" would be taken into account individually, this would lead to the negation of the meaning of the moview review. This could be avoided by implementing a bigram or trigram model. By first implementing the a bigram model run time dramatically increased because of massive scaling in data and the accuracy decreased to 80.2% with 67% accuracy for the negative class and 94% for the positive class. After some reassessing the program was rerun with the removal of the stopwords. This increased the the accuracy to its all time high of 85.35%. Increasing the accuracy of the negative class but decreasing the it of the positive class. See image below with stopwords removed and stopwords not removed.



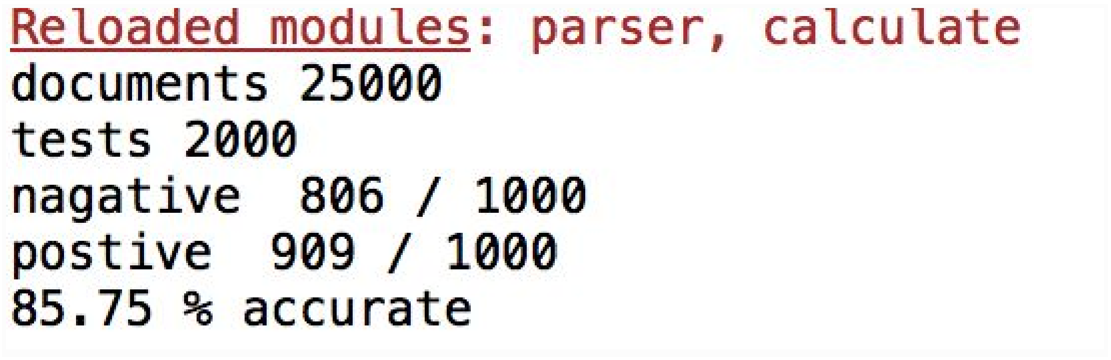
```
Reloaded modules: parser, calculate
documents 25000
tests 2000
nagative  667 / 1000
postive  937 / 1000
80.2 % accurate

In [3]: runfile('/Users/alexkirwan/College/
machineLearning/dev/BayesianClassification_
main.py', wdir='/Users/alexkirwan/College/
machineLearning/dev/BayesianClassification_
Reloaded modules: parser, calculate
documents 25000
tests 2000
nagative  783 / 1000
postive  924 / 1000
85.35 % accurate
```

The trigram model was then tested without the removal of stopwords. This had the largest negative impact on the accuracy dropping the average accuracy to 78.0%, with the majority of the accuracy lost in the negative category.



```
Reloaded modules: parser, calcul
documents 25000
tests 2000
nagative  594 / 1000
postive  966 / 1000
78.0 % accurate
```

The program was then reverted back back to the bigram model and by also implementing the nltk tokenizer on the testing data also a small boost of .35%



```
Reloaded modules: parser, calculate
documents 25000
tests 2000
nagative  806 / 1000
postive  909 / 1000
85.75 % accurate
```

Stemming was implemented once again this time with a bigram model and with stopwords making 0.00% difference which is a big change from the from the 1.2% lost when trying stemming without the n-gram model. Proving the importance of context.

Overall the highest accuracy recorded was 85.75% this was achieved by using a bigram model without removing stopwords all on a clean dataset of token not including special characters or punctuation. The negative accuracy was always around 10% less than that of the positive class. From research the 90% accuracy is achievable by using further pre processing techniques such as fisher’s least significant difference, acronym and emoticon replacement or other methods of dealing with negation. The naive bayes is a very simple and straightforward algorithm that with the correct pre processing techniques can be very accurate and one of the best classification models being used today.