NEWSPAPER CATEGORY CLASSIFIER

INTRODUCTION

Machine learning can be divided into two categories; supervised learning and unsupervised learning. With supervised learning, a model learns from a training set that is already labelled, with examples like classification and regression. Unsupervised learning however, discovers hidden structure in homogenous groups that have similar characteristics. The training data are not labelled and examples include clustering and graphical models.

The purpose of this project is to write a classifier to address the document classification problem. It is a document classification problem as the response variable is already provided and is a discrete set of labels. The classifier needs to be able to correctly predict the category label of a given article by analysing the text. The solution focuses on the understanding of feature generation and choosing the appropriate classification model to implement. The feature generation Bag of Words model and the following classification models will be implemented to resolve this problem; Bernoulli Naïve Bayes, Gaussian Naïve Bayes, Linear Support Vector Classifier, Multinomial Naïve Bayes, Random Forest and Support Vector Classifier – RBF Kernel.

METHOD

For the classifiers that we will implement to address this document classification problem, a Bags of Words model will be used for the feature generation. Within this model, a text (such as a document or sentence) is characterised as the bag of its words, neglecting grammar, stop words, word order but maintains the multiplicity. Stop words refer to the most common words in a language. The frequency of each word is used as a feature for training a classifier. Thus, this shall help improve the accuracy of our classifiers. The classifiers that we will implement and test for this document classification problem are the Bernoulli Naïve Bayes, Gaussian Naïve Bayes, Linear Support Vector Classifier, Multinomial Naïve Bayes, Random Forest and Support Vector Classifier – RBF Kernel.

The Bernoulli Naïve Bayes classifier uses binary term occurrence features instead of term frequencies and is considered a popular model for document classification problems. In this model, the features are independent Booleans (binary variables) describing inputs. The Gaussian Naïve Bayes model however, is similar to the previous model, with the exception that the features of the input vector are continuous instead of discrete. In contrast to this, the Multinomial Naïve Bayes model is an event model that has events representing the occurrence of a word in a single document. With this model, it is commonly used for document classification.

In theory, out of the three Naïve Bayes model, the Multinomial Naïve Bayes model would be the most appropriate and most accurate for this document classification problem. The Multinomial Naïve Bayes model uses term frequencies which is ideal in this case, whereas the Bernoulli Naïve Bayes model uses binary term occurrence features. The Gaussian Naïve Bayes model is not suitable for this problem as the model takes continuous features where discrete features are required. We will implement all three of these models with the Bag of Words model to see whether or not that the Multinomial Naïve Bayes model is the most appropriate out of the three. For each model, three test runs will be performed with the average of the three results taken for the predictor accuracy.

In terms of other supervised learning models, we will implement and test a few support vector machines (SVMs) such as the Linear Support Vector Classifier and the Support Vector Classifier – RBF Kernel. An SVM training algorithm constructs a model that allocates new examples into one category or another, making it a non-probabilistic binary linear classifier. An SVM model represents examples as points in space in a way where they are mapped so that separate categories are divided by clear gaps. Dependent on which side of the gaps that the new examples fall on, this will form the basis for the prediction of which category they belong to. Besides performing linear classification, SVMs can efficiently perform a non-linear classification using something called a kernel trick; implicitly mapping their inputs into high dimensional feature spaces.

From the two SVM models mentioned above, the Linear Support Vector Classifier seems the most fitting for this document classification task as it is a linear problem. The Support Vector Classifier – RBF Kernel model will not perform as well as the Linear Support Vector Classifier for this problem as this model is primarily used for non-linear classification problems. We will implement both of these models along with the Bags of Words model and see if the Linear Support Vector Classifier does indeed outperform the Support Vector Classifier – RBF Kernel model. For each model, three test runs will be performed with the average of the three results taken for the predictor accuracy.

Lastly, the final model we will implement and test is the Random Forest model. The Random Forest model is suitable for this problem as it is an extension of decision trees; designed to achieve better predictions and prevent overfitting. The Decision Tree model, builds decision trees around explanatory variables that are considered the most distinguishing features for pre-defined classes. The idea behind the Random Forest model is to produce an entire forest of decision trees where each individual one is trained on a randomly chosen subset of data. Combining these trees into a forest, should increase the accuracy of the final tree. We shall implement the Random Forest model along with the Bags of Words model and assess its performance compared to the supervised

learning models discussed earlier. We will perform three test runs for this model and will take the average of the three results for the predictor accuracy.

RESULTS

Running the Bernoulli Naive Bayes, Gaussian Naïve Bayes and Multinomial Naïve Bayes models produced the following:

```
In [3]: %run ./test.py -b
Predictor: Bernoulli Naive Bayes
Number of correct predictions: 983
Number of incorrect predictions: 1017
Total predictions: 2000
Predictor Accuracy: 49.15%

In [4]: %run ./test.py -b
Predictor: Bernoulli Naive Bayes
Number of correct predictions: 983
Number of incorrect predictions: 1017
Total predictions: 2000
Predictor Accuracy: 49.15%

In [5]: %run ./test.py -b
Predictor: Bernoulli Naive Bayes
Number of correct predictions: 983
Number of incorrect predictions: 1017
Total predictions: 2000
Predictor Accuracy: 49.15%
```

Figure 1: Bernoulli Naïve Bayes

```
In [6]: %run ./test.py -g
Predictor: Gaussian Naive Bayes
Number of correct predictions: 1008
Number of incorrect predictions: 992
Total predictions: 2000
Predictor Accuracy: 50.4%

In [7]: %run ./test.py -g
Predictor: Gaussian Naive Bayes
Number of correct predictions: 1008
Number of incorrect predictions: 992
Total predictions: 2000
Predictor Accuracy: 50.4%

In [8]: %run ./test.py -g
Predictor: Gaussian Naive Bayes
Number of correct predictions: 1008
Number of incorrect predictions: 992
Total predictions: 2000
Predictor Accuracy: 50.4%
```

Figure 2: Gaussian Naïve Bayes

```
In [15]: %run ./test.py -m
Predictor: Multinomial Naive Bayes
Number of correct predictions: 1295
Number of incorrect predictions: 705
Total predictions: 2000
Predictor Accuracy: 64.75%

In [16]: %run ./test.py -m
Predictor: Multinomial Naive Bayes
Number of correct predictions: 1295
Number of incorrect predictions: 705
Total predictions: 2000
Predictor Accuracy: 64.75%

In [17]: %run ./test.py -m
Predictor: Multinomial Naive Bayes
Number of correct predictions: 1295
Number of incorrect predictions: 705
Total predictions: 2000
Predictor Accuracy: 64.75%
```

Figure 3: Multinomial Naïve Bayes

As we can see, our hypothesis was correct with the Multinomial Naïve Bayes model being the most suitable and accurate out of the three models. The Bernoulli Naïve Bayes and Gaussian Naïve Bayes models produced similar results as we can see, with both around the 50% accuracy mark. The Multinomial Naïve Bayes model greatly outperforms the other two with an accuracy rate of 64.75%.

Running the Linear Support Vector Classifier and the Support Vector Classifier – RBF Kernel models, showed the following:

```
In [12]: %run ./test.py -l
Predictor: Linear Support Vector Classifer
Number of correct predictions: 1286
Number of incorrect predictions: 714
Total predictions: 2000
Predictor Accuracy: 64.3%

In [13]: %run ./test.py -l
Predictor: Linear Support Vector Classifer
Number of correct predictions: 1286
Number of incorrect predictions: 714
Total predictions: 2000
Predictor Accuracy: 64.3%

In [14]: %run ./test.py -l
Predictor: Linear Support Vector Classifer
Number of correct predictions: 1286
Number of incorrect predictions: 714
Total predictions: 2000
Predictor Accuracy: 64.3%
```

Figure 4: Linear Support Vector Classifier

```
In [22]: %run ./test.py -s
Predictor: Support Vector Classifer - RBF Kernel
Number of correct predictions: 881
Number of incorrect predictions: 1119
Total predictions: 2000
Predictor Accuracy: 44.05%

In [23]: %run ./test.py -s
Predictor: Support Vector Classifer - RBF Kernel
Number of correct predictions: 881
Number of incorrect predictions: 1119
Total predictions: 2000
Predictor Accuracy: 44.05%

In [24]: %run ./test.py -s
Predictor: Support Vector Classifer - RBF Kernel
Number of correct predictions: 881
Number of incorrect predictions: 1119
Total predictions: 2000
Predictor Accuracy: 44.05%
```

Figure 5: Support Vector Classifier – RBF Kernel

From the results, we can see that the Linear Support Vector Classifier clearly outperforms the Support Vector Classifier – RBF Kernel by 20%. This fits our original theory that the Support Vector Classifier – RBF Kernel would not perform as well as the Linear Support Vector Classifier as this is a linear classification problem.

Last of all, the Random Forest model with 200 trees produced the following results:

```
In [19]: %run ./test.py -r
Predictor: Random Forest
200 trees

Number of correct predictions: 1304
Number of incorrect predictions: 696
Total predictions: 2000
Predictor Accuracy: 65.2%

In [20]: %run ./test.py -r
Predictor: Random Forest
200 trees

Number of correct predictions: 1287
Number of incorrect predictions: 713
Total predictions: 2000
Predictor Accuracy: 64.35%

In [21]: %run ./test.py -r
Predictor: Random Forest
200 trees

Number of correct predictions: 1286
Number of incorrect predictions: 714
Total predictions: 2000
Predictor Accuracy: 64.3%
```

Figure 6: Random Forest (200 trees)

From the output we can see that the Random Forest (200 trees) model produced a similar level of accuracy to the Multinomial Naïve Bayes (Figure 3) and the Linear Support Vector Classifier (Figure

4) models. Modifying the number of trees to 100 and 300 for the Random Forest model, produced the following:

```
In [2]: %run ./test.py -r
Predictor: Random Forest
100 trees

Number of correct predictions: 1292
Number of incorrect predictions: 708
Total predictions: 2000
Predictor Accuracy: 64.60000000000001%

In [3]: %run ./test.py -r
Predictor: Random Forest
100 trees

Number of correct predictions: 1273
Number of incorrect predictions: 727
Total predictions: 2000
Predictor Accuracy: 63.65%

In [4]: %run ./test.py -r
Predictor: Random Forest
100 trees

Number of correct predictions: 1300
Number of incorrect predictions: 700
Total predictions: 2000
Predictor Accuracy: 65.0%
```

Figure 7: Random Forest (100 trees)

```
In [5]: %run ./test.py -r
Predictor: Random Forest
300 trees

Number of correct predictions: 1316
Number of incorrect predictions: 684
Total predictions: 2000
Predictor Accuracy: 65.8%

In [6]: %run ./test.py -r
Predictor: Random Forest
300 trees

Number of correct predictions: 1300
Number of incorrect predictions: 700
Total predictions: 2000
Predictor Accuracy: 65.0%

In [7]: %run ./test.py -r
Predictor: Random Forest
300 trees

Number of correct predictions: 1300
Number of incorrect predictions: 700
Total predictions: 2000
Predictor Accuracy: 65.0%
```

Figure 8: Random Forest (300 trees)

The results are quite similar, with the exception of having a lower number of trees producing a slightly less accurate prediction and with a higher number of trees producing a slightly higher prediction accuracy.

Overall, the results showed that the lower performing classification models for this document classification problem were the Bernoulli Naïve Bayes, Gaussian Naïve Bayes and the Support Vector Classifier – RBF Kernel models with prediction accuracy ranging from 44% - 50%. The top performing classification models were the Multinomial Naïve Bayes, Linear Support Vector Classifier and the Random Forest with prediction accuracy of around 64% - 65%.

CONCLUSION

The project presented a document classification problem that entailed predicting the category of a given article from the New York Times. The Bag of Words model was implemented for feature generation and was then used for the following supervised learning classification models: Bernoulli Naïve Bayes, Gaussian Naïve Bayes, Linear Support Vector Classifier, Multinomial Naïve Bayes, Random Forest and Support Vector Classifier – RBF Kernel. The outcome of this project revealed that the lowest performing classification models for this problem were the Bernoulli Naïve Bayes, Gaussian Naïve Bayes and the Support Vector Classifier – RBF Kernel; ranging from 44% - 50%

prediction accuracy. The top performing classification models were the Linear Support Vector Classifier, Multinomial Naïve Bayes and the Random Forest with prediction accuracies of 64.3%, 64.75% and 65%.

REFERENCES

http://easymachinelearning.tumblr.com/post/51332575179/gaussian-naive-bayes-classifier

https://en.wikipedia.org/wiki/Bag-of-words_model

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

https://en.wikipedia.org/wiki/Support_vector_machine