MiniProject 2: Classification of Textual Data

COMP 551, Winter 2022, McGill University

Manas Kale	**Gauri Sharma**	**Avinash Bhat**
261000867	261026894	260969537

Abstract

This mini-project is aimed at studying the application of machine learning algorithms to the task of classifying textual data. Our study was based on two datasets namely, 20 News group dataset and Sentiment140 dataset. Both of these datasets comprise of collections of text documents and their respective labels. We performed a multi-class classification on the 20 News group dataset and binary classification on the Sentiment140 dataset. Scikit-Learn python package was used for loading the data, feature extraction and feature selection. We implemented Naive Bayes and Logistic Regression models which were trained using our implementation of 5-folds cross validation. We observed that the Naive Bayes model outperformed the Logistic Regression model for both the datasets.

1 Introduction

Text classification is one of the fundamental problem solved using machine learning techniques. Text classification is the task of classifying a document under an already defined category (Jindal et al., 2015). One of the major tasks in text classification is the assignment of labels to text, based on its characteristics. This particular task has broad applications, for example, topic detection, spam classification, sentiment analysis etc.
Single-label is when a document is assigned to only one class whereas if it is assigned to more that one class it is called *multi-label* (Kowsari et al., 2019). Humans can perform classification tasks pretty well since they analyze the world for describing various objects in it. In the recent times the amount of information has increased, therefore, in order to categorize a significantly large amount of textual data we need automated text classification, otherwise it can get time-consuming for human counterparts. This project asks us to perform classification on two datasets - 20 News group and Sentiment 140 dataset.
We performed multi-label classification on 20 News group and binary classification on Sentiment140 using two different classifiers - Naive Bayes and logistic regression. To train our models and find the best hyperparameters, we used 5-fold cross validation. All 3 of these (Naive Bayes, logistic regression and cross validation) were implemented from scratch. We also implemented Grid Search from scratch to evaluate the hyperparameters for both the algorithms.

2 Datasets

2.1 20 News group dataset

The 20 news group dataset comprises of 18846 news text posts on 20 different topics (eg., ”graphics”, ”space”, ”guns” etc.) split in two subsets: one for training and the other one for testing. The split between the train and test set is based on the date when a news article was posted i.e., all posts before this date are for training and posts after are for testing. On performing exploratory analysis (Figure 1, 2), we discovered that this dataset has relatively evenly distributed training and testing samples.

2.1.1 Data Processing

Since this is textual data, we applied the following text preprocessing steps:

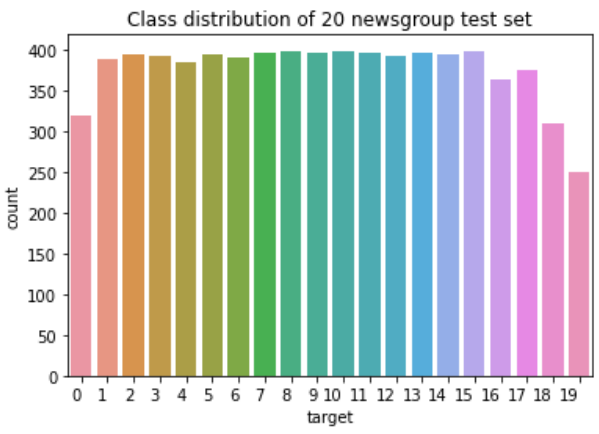

Figure 1: class distribution for 20News test

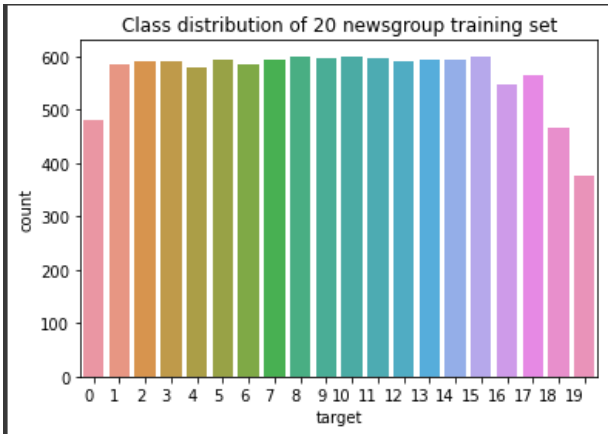

Figure 2: class distribution for 20News train

- **Special character removal** : Special characters are usually not associated with any news category and might confuse the classifier, so these were removed. We also removed leading and trailing whitespaces.
- **Lowercase conversion** : Upper and lowercase letters have the same semantic meaning, so all letters were convereted to lowercase.
- **Stop-word removal**: Stop-words such as ”the” carry no semantic meaning so were removed using NLTK’s standard English stop-word dictionary.

Additionally, we hypothesized that single words may not encode a class correctly. For example, *rocket* and *space rocket* belong to two completely different classes - *talk.politics.guns* and *sci.space*. Since there are many such examples in the English language, we used bigram word features. To make sure rare words were accounted for, we tried using Term Frequency - Inverse Document Frequency (TF-IDF) features as well.

2.1.2 Training, validation and testing sets

We used the built-in train/test split, giving us 11314 samples for testing and 7532 for training. The training set was further truncated by randomly selecting 20%, 40%, 60%, 80% (Figure 8) and split into validation and training sets as per 5-folds cross validation algorithm. Before deciding fold indices, the training samples were also randomized to diminish effect of skewed data.

2.2 Sentiment140 dataset

This is a relatively large dataset used for sentiment analysis. The training data, which has around 1,600,000 instances, was gathered by marking tweets with positive emoticons as positive and those with negative emoticons as negative. The training dataset in comparison is relatively smaller. We found that although the train set had only 2 classes (positive and negative), the testing set had 3 (positive, negative and neutral). We disregarded the neutral class in our experiments. Looking at the class distribution (Figure 3), we see that the training data is relatively balanced.

2.2.1 Data Processing

Since this is text data as well, the same data processing steps as used for 20News were performed here. Here the removal of special characters is important because the training data was labelled based on emoticons. This could potentially train the model such that it associates emoticons with sentiment, which will be an issue if we try to use this model to predict sentiment of text larger than tweets. Unlike tweets, people generally tend to express sentiments in sentences through meaningful words. However, we ran into memory issues while generating ngrams. We tried to load the data into the memory using the following techniques. We used the HashingVectorizer function provided by sklearn. This is a good

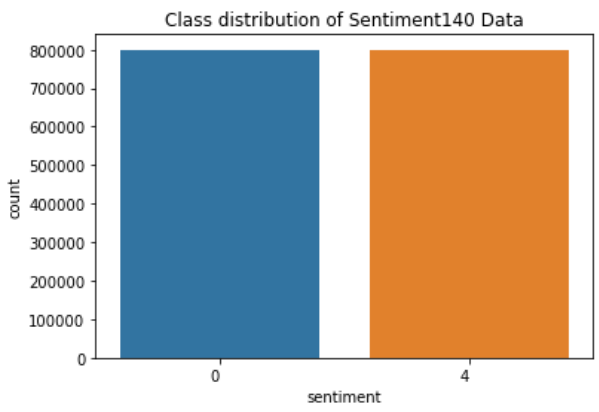

Figure 3: class distribution for sentiment140 training set with 0=negative, 4=positive

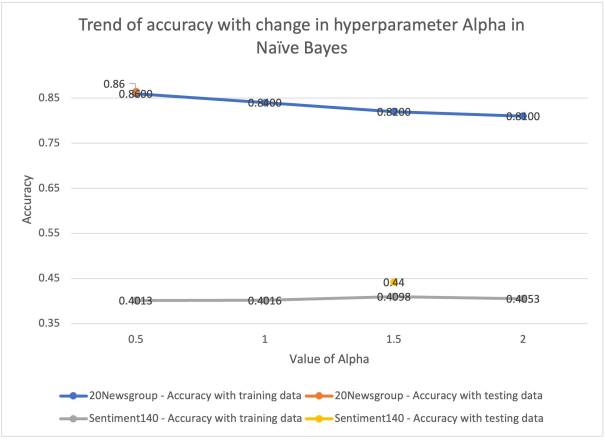

Figure 4: Hyperparameter tuning for the Naïve Bayes dataset

technique to load the data since it does not store the vocabulary in memory, but instead stores their hashes. However, this technique did not work since the data was too large. We also tried truncating the sentences (in order to limit the feature space). However, the average sentence length was not too long and in many cases, this resulted in the sentence length of a data point to be zero. We then reduces the dataset gradually from 100%, 70%, 50%, 30% and then finally 10% of the original dataset. 10% of the original dataset ran without any issues (except for the trigrams).

2.2.2 Training, validation and testing sets

The training set was further truncated by randomly selecting 10% of the data and split into validation and training sets as per 5-folds cross validation algorithm. Before deciding fold indices, the training samples were also randomized to diminish effect of skewed data.

3 Results

For both the datasets, we extracted the character count and the word count, before and after preprocessing the text. However, we did not notice a good correlation between these features and the target variable in both the datasets. Next we generated the bigrams and trigrams and the TF-IDF from the text data. Trigrams take up huge memory and caused OOM issues for many experiment runs (especially for the Sentiment140 dataset). We were able to run our experiments successfully using the bigrams and TF-IDF features. Overall we found that the TF-IDF features performed better so we compare our models based on this feature.

We were able to run experiments comparing the different hyperparameters for the naive bayes and the logistic regression models. For Naive Bayes, we found that the **alpha value of 1.5** gave the best training and testing accuracy.

We experimented with the different values of C, maximum iterations and learning rates for the logistic regression model. The results are shown in the figure 7. Overall we found that the **learning rate of 0.1, maximum iteration value 50 and C value 0.2** worked well for both the datasets.

Finally, we experimented with the data size for both the datasets for both the models as shown in figure 8 and figure 9.

Overall we find that the Naive Bayes Model performed better for both the datasets (using the best hyperparameters found using Grid Search). The results are tabulated in table 1.

4 Discussion and Conclusion

We explored text classification using naive bayes and softmax regression models. Testing accuracies were heavily influenced by the type of feature encoding that was used (i.e, bag of words, bigrams, trigrams etc.). We observed that it is important to find the correct trade-off between resource availability

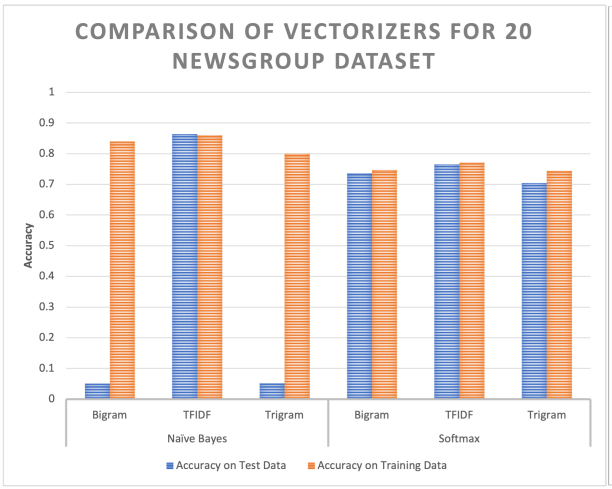

Figure 5: Comparison of vectorizers for 20 News group dataset

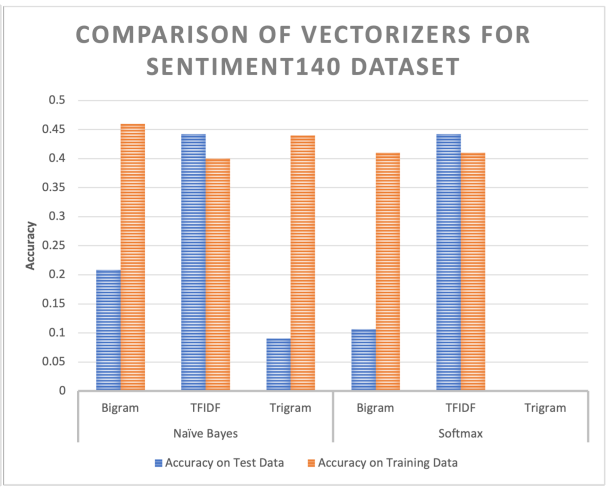

Figure 6: Comparison of vectorizers for Sentiment 140 dataset

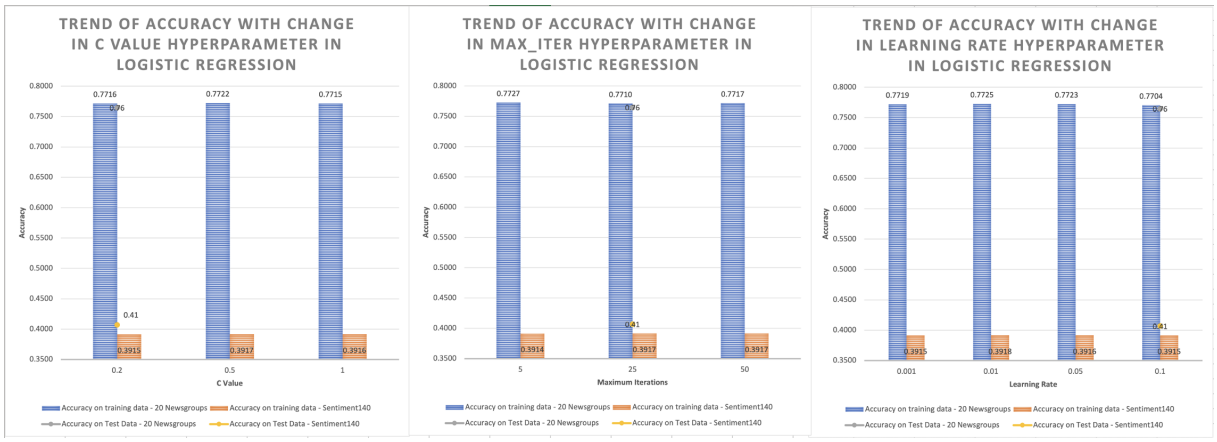

Figure 7: Train/test accuracies for various Logistic regression hyperparameters

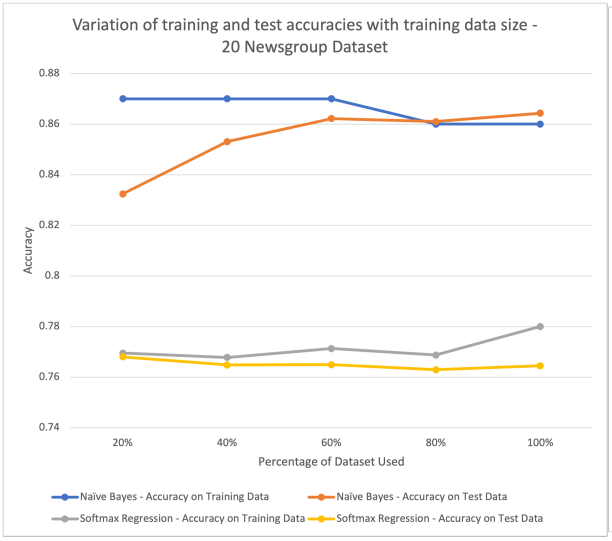

Figure 8: Variation of training and test accuracies with datasize for 20 Newsgroup dataset

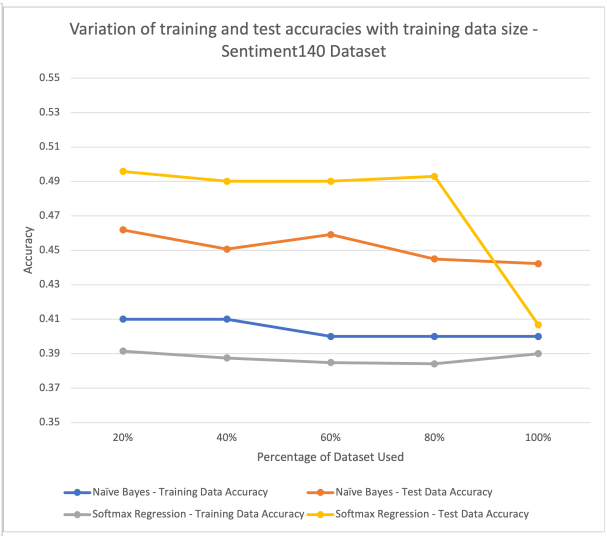

Figure 9: Variation of training and test accuracies with datasize for Sentiment140 dataset

	Sentiment140 Dataset		20 Newsgroup	
Algorithms	**Training Accuracy**	**Test Accuracy**	**Training Accuracy**	**Test Accuracy**
Naïve Bayes	0.4	**0.4422**	0.86	**0.8643**
Logistic Regression	0.3915	0.4067	0.7723	0.7645

Table 1: Results of Naïve Bayes and Logistic Regression models for both the datasets

and feature complexity since n-grams on the larger dataset consume a lot of memory for very little gain in accuracy.
For binary classification tasks especially, support vector machine models might give better performance since they learn decision boundaries. It would also be interesting to use algorithms like word2vec as feature embeddings for the classifier as they can encode some model of word ”meanings” in feature vectors. This could also help in complicated sentiment classification examples such as sarcasm.

5 Statement of Contributions

Avinash implemented Logistic Regression, ran experiments and worked on this report.
Gauri did Naive Bayes, data cleaning and processing, exploratory data analysis and report writing.
Manas implemented cross validation and worked on report writing.

References

Rajni Jindal, Ruchika Malhotra, and Abha Jain. 2015. Techniques for text classification: Literature review and current trends. *Webology*, 12.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. 2019. Text classification algorithms: A survey. *Information*, 10(4).