COMP 551 Mini Project 2

Lansu Dai, Zichuan Guan, and Lan Xi Zhu

Abstract. We investigated the performance of Logistic regression and Naive Bayes on natural language classification tasks - binary and multi-class text classifications. Using 5-fold cross-validation, we found that different preprocessing techniques, such as lemmatization and stemming, do not always improve the performance of a classifier. We also observed that multinomial Naive Bayes performed better than Gaussian Naive Bayes. Both Logistic regression and Naive Bayes models performed better in binary classification, with an accuracy of 0.730 and 0.735 respectively, while having an accuracy of 0.691 and 0.633 in multi-class classification. We also investigated their performance relative to the training size. Both models showed significant improvement when the data size increases. Multinomial Naive Bayes' performance increase is generally bigger than that of the Logistic regression when more data is given.

1 Introduction

The main project task is to implement the naive Bayes (NB) classifier and to apply it on textual datasets. The NB model, implemented from scratch, is then compared with the logistic regression classifier from the scikit-learn package for their performances on two different datasets. K-fold cross-validation is used for model selection and hyperparameter tuning.

The 20 Newsgroups dataset (20NG), with textual instances categorized into twenty topics depending on their contents, is contained in the scikit-learn package and is a popular dataset in machine learning [1]. It is widely used in benchmarking text cleaning techniques [2], text categorization algorithms [3], and many other fields related to natural language processing. The Sentiment140 dataset (S140) is a collection of Tweets, with each textual instance classified by the polarity (either positive or negative) that it expresses [4]. Being extracted from social media, it can be used in testing algorithms that monitor public opinion [5] or generate content recommendation [6].

Our results show that multinomial Naive Bayes classifier, the better out of the two experimented models (multinomial likelihood and Gaussian likelihood), reaches a test accuracy of 0.735 in the binary S140 dataset, and 0.633 in the multi-class 20NG dataset. The logistic regression classifier has a lower test accuracy (0.730) than the former in S140 dataset, but performs better in the 20NG dataset, with an accuracy of 0.691.

2 Datasets

As previously mentioned, the instances in 20NG belong to twenty content categories. The 11314 instances in the training set are relatively evenly distributed amongst the twenty classes, with a maximum of 600 instances per class and a minimum of 377. This dataset has a relatively large test set with 7532 instances.

As for S140, the 1,600,000 instances in the training set contain exactly 800,000 positive labels and 800,000 negative labels, with a much smaller test set of 359 texts after removing all instances with neutral labelling. To avoid memory issues, we randomly pick a training subset of 10,000 instances (4997 positives, 5003 negatives) to train our models.

Before tokenizing the textual instances, we preprocess them by removing non-informative features like email addresses (with @s), headers (e.g. From article by), and website links. We use scikit-learn's built-in stop word list for English for our purpose. We attempt two different tokenizers using NLTK stemmers, besides scikit-learn's default tokenization step. The SnowballStemmer strips suffix from word features (e.g. connection and connective are both mapped to connect) [7]. In practice, stemming might result in spelling errors, because it simply removes the suffix (e.g. stemming becomes stemm) without considering the context. The WordnetLemmatizer, on the other hand, converts word features to their base form according to the Wordnet English lexical database [8]. All aforementioned preprocessing steps aim to reduce the number of word features (tokens) in our further analysis by ignoring the non-informative words and handling repetitive features (e.g. words with identical lemma).

3 Results

3.1 Naive Bayes

Two different vectorizers are used to generate features based on the given textual instances. The CountVectorizer converts texts to a sparse matrix of token counts [9]. Since we have discrete integer counts as input, multinomial likelihood is assumed in the corresponding Naive Bayes classifier. As an alternative, we also use Gaussian Naive Bayes classifier with TfidfVectorizer, which gives the term-frequency (scaled by inverse document-frequency, assuming the frequently occurring words to be less informative) of each word feature instead of the raw counts. These values are assumed to be continuous and to follow a Gaussian distribution. We use 5-fold cross-validation to choose between these two Naive Bayes classifiers.

For 20NG, our multinomial classifier achieves its best validation accuracy, 0.670, using CountVectorizer with English stop words. The Gaussian classifier achieves its best validation accuracy, 0.631, using TfidfVectorizer with English stop words.

For S140, the former has a validation accuracy of 0.730 (default CountVectorizer), whereas the latter has a validation accuracy of 0.592 (default TfidfVectorizer). We observe that for both datasets, the multinomial Naive Bayes classifier performs better in 5-fold cross-validation and is thus chosen to represent the Naive Bayes model.

3.2 Logistic Regression

The LogisticRegression method in the scikit-learn package takes multiple parameters. With both datasets, we run 5-fold cross-validation through four vectorizer settings [1], four solvers (*newton-cg*, *lbfgs*, *sag*, *saga*) and three inverse regularization strength (10, 1, 0.1). All other parameters are kept constant throughout the validation process.

[1]Four variations of CountVectorizer (TfidfTransformer is applied subsequently, making it equivalent to TfidfVectorizer): 1) with default parameters, 2) with English stop words, 3) with English stop words + SnowballStemmer, 4) with English stop words + WordnetLemmatizer.

Dataset \ Model	Naive Bayes	Logistic Regression
Sentiment140	0.735	0.730
20 Newsgroups	0.633	0.691

Table 1: A comparison of the test accuracy of Naive Bayes (multinomial likelihood) and logistic regression on both datasets.

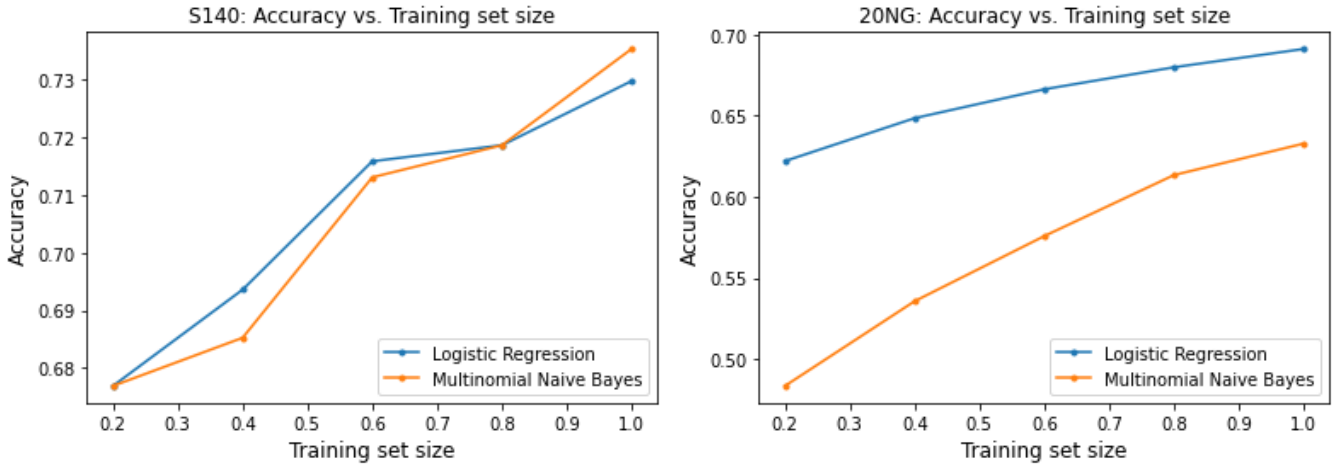

Figure 1: Test accuracy vs. training set sizes for S140(left) and 20NG(right) datasets using logistic regression (with the best parameters for either datasets) and multinomial Naive Bayes.

For 20NG, the combination with the highest validation accuracy (0.755) is determined to be: CountVectorizer with English stop words, *saga* solver, and an inverse regularization strength of 10. The best parameters found for S140 are: default CountVectorizer, *saga* solver, and no regularization ($C = 1$). This yields a validation accuracy of 0.746. Due to hardware limitation, we kept a maximum number of 100 iterations when fitting, only the *lbfgs* solver suffered convergence problem.

While we expect that preprocessing steps such as stemming or lemmatizing to improve model performance, cross-validation performed on both models show that CountVectorizer with English stop words works the best for 20NG, whereas the default CountVectorizer works the best for S140.

3.3 Comparison

The comparison of test accuracy between Naive Bayes (multinomial likelihood) and logistic regression classifiers are shown in Table 1. The Naive Bayes classifier performs better with the S140 dataset, whereas the logistic regression classifier performs better with the 20NG dataset. The effect of training set size on model performance is also investigate and the results are shown in Figure 1.

4 Discussion and Conclusion

In the context of text categorization, in particular for the two datasets we explore in this project, the number of features (i.e. distinct tokens) can easily reach 10^5 or more. Thus, the efficiency of our classifying algorithms becomes very important. With limited RAM capacity, we are only able to train our classifiers

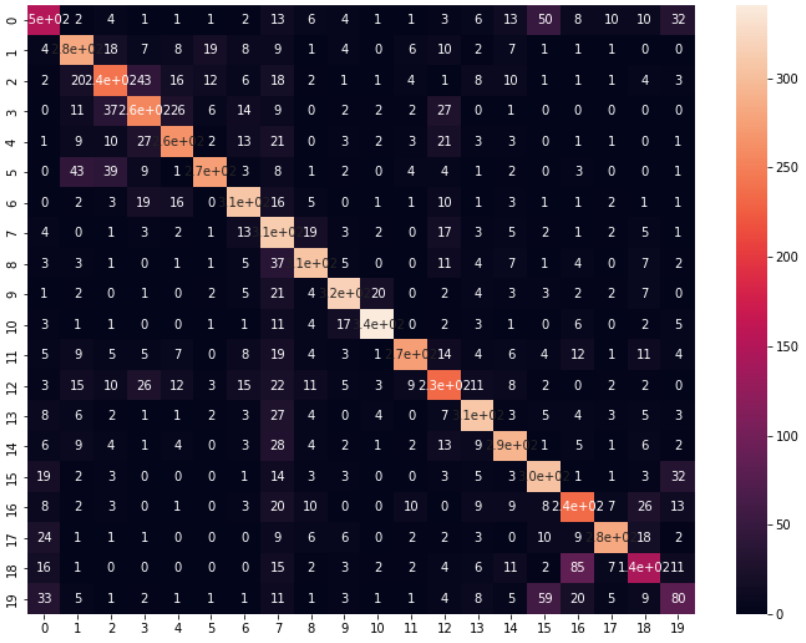

Figure 2: Confusion matrix for 20NG dataset using logistic regression with the best parameters.

with a small subset of S140 training set. We expect the classifiers to have better performance with larger datasets. As shown in Figure 1, test accuracy strictly increases with training set size for both models with both datasets. In particular, the test accuracy of the multinomial Naive Bayes model is greatly influenced by the training set size. With steeper slopes in both plots, the Naive Bayes model is outperformed by Logistic regression when the dataset is small, and it gradually catches up when the training size increases.

Looking at the confusion matrix in Figure 2, we observe that the two classes with the highest confusion rate is class 16 (talk.politics.guns) and class 18 (talk.politics.misc), which talk about similar topics. Class 15 (soc.religion.christian) is also frequently mixed up with Class 19 (talk.religion.misc) and Class 0 (alt.atheism). These three classes are all religious-related. The classifiers would likely benefit from more training instances from these classes that are similar content-wise.

To conclude, our results show that both Naive Bayes and logistic regression classifiers, with part of their associated parameters tuned via 5-fold cross-validation, perform better in binary classification tasks as compared to multi-class classification given the dataset size within the scope of this project. In particular, the test accuracy of multinomial Naive Bayes classifier tops that of logistic regression classifier in the S140 dataset. The opposite happens for the 20NG dataset. The performance of both models improves while more textual instances are given as training set.

Statement of Contributions

Everyone contributed equally to the coding component and the project write-up. We have not assigned specific part of the project to individual group member.

References

[1] K. Lang, "Newsweeder: Learning to filter netnews," in *Machine Learning Proceedings 1995*, pp. 331–339, Elsevier, 1995.

[2] K. Albishre, M. Albathan, and Y. Li, "Effective 20 newsgroups dataset cleaning," in *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, vol. 3, pp. 98–101, IEEE, 2015.

[3] Z. Elberrichi, A. Rahmoun, and M. A. Bentaalah, "Using wordnet for text categorization.," *International Arab Journal of Information Technology (IAJIT)*, vol. 5, no. 1, 2008.

[4] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N project report, Stanford*, vol. 1, no. 12, p. 2009, 2009.

[5] B. S. Shekhawat, *Sentiment Classification of Current Public Opinion on BREXIT: Naïve Bayes Classifier Model vs Python's TextBlob Approach*. PhD thesis, Dublin, National College of Ireland, 2019.

[6] K. Gandhe, A. S. Varde, and X. Du, "Sentiment analysis of twitter data with hybrid learning for recommender applications," in *2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 57–63, IEEE, 2018.

[7] M. F. Porter, "Snowball: A language for stemming algorithms," 2001.

[8] Princeton University, "About wordnet.," 2010.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.