551 MiniProject2

Fynn Schmitt-Ulms, Joon Hwan Hong, and Daniel Korsunsky

February 28, 2021

Abstract

In this project, we investigated the performance of two linear classification models (Naive Bayes and logistic regression) with 5-fold cross validation on two benchmark datasets: the 20 Newsgroup dataset from scikit-learn [1] and the IMDb reviews dataset from Andrew Maas at Stanford [2]. A numpy RNG seed of '551' was set for this exploration. We found the optimal Naive Bayes hyperparameters on both datasets to be `laplace_val` = 1 and `log_correction` = 0.01, while the optimal Logistic Regression hyperparameter was `C` = 10 for the IMDb dataset and `C` = 100 for the Newsgroups dataset. With these optimal hyperparameters, the Naive Bayes model outperformed the Logistic Regression model on the Newsgroup test accuracy, while the Logistic Regression model outperformed the Naive Bayes model on the IMDb test accuracy. For the Newsgroup dataset, the impact of reduced training data availability was greater compared to the IMDb dataset. In an additional experiment, introducing n-grams of size ≤ 2 (word-pair features) achieved slightly better test accuracy on the IMDb dataset, but slightly worse test accuracy on the Newsgroup dataset.

1 Introduction

The 20 Newsgroup and IMDb datasets used in this analysis have been used in the past to develop machine learning methodologies. Adi and Çelebi (2014) used the Newsgroup dataset to test a naive Bayes classifier against other classifiers [3], while Maas et al. (2011) used the IMDb dataset to develop unsupervised and supervised techniques for various NLP tasks. Our work uses K-fold cross validation to determine the best naive Bayes and logistics regression hyperparameters for the text datasets (Exp. 1), as well as compare the performances of these optimized models across datasets (Exp. 2) and as a function of dataset size (Exp. 3).

The naive Bayes model was implemented as a class with a `fit` and `predict` function, which matches the design of the scikit-learn classifiers. In addition, an `evaluate_acc` function was defined to evaluate model accuracy. The logistic regression model was taken from the `scikit-learn` package. In our hyperparameter search with 5-Fold cross-validation (implemented from scratch), we found the optimal Naive Bayes hyperparameters on both datasets to be `laplace_val` = 1 and `log_correction` = 0.01, while the optimal Logistic Regression hyperparameter was `C` = 10 for the IMDb dataset and `C` = 100 for the Newsgroups dataset. Log correction is a hyperparameter we added to the Naive Bayes model to prevent numerical errors. Similar to Laplace smoothing, we add the small log correction constant to both the numerator and denominator when calculating the Naive Bayes θ parameters. This prevents $log(0)$ from being calculated during the prediction stage when a feature never occurs with a class label.

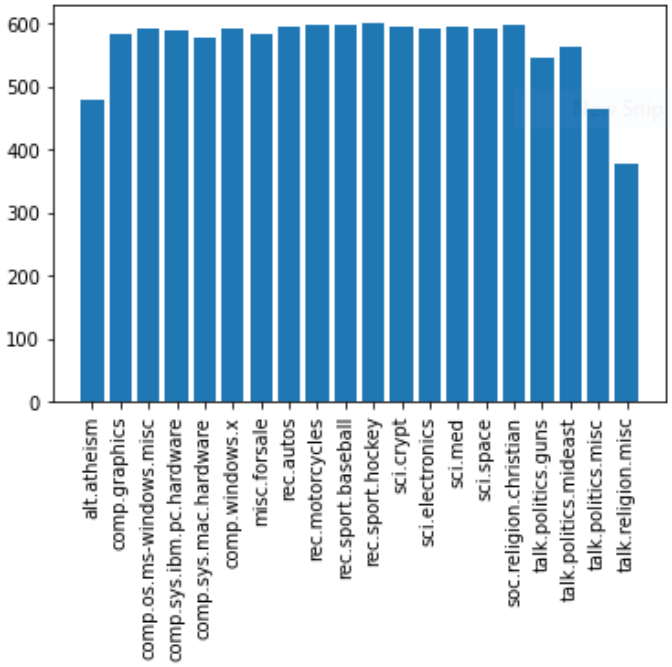
With these optimal hyperparameters, the Naive Bayes model outperformed the Logistic Regression model on the Newsgroup test accuracy, 0.6686 to 0.6168. For the IMDb test accuracy, the Logistic Regression model outperformed the Naive Bayes model, 0.8832 to 0.8393. For the 20 Newsgroup dataset, the impact of reduced training data availability was greater compared to the IMDb dataset. For example, for the Naive Bayes model, there was far greater difference between the 20% accuracy and 80% (0.5358 vs. 0.6563) in the Newsgroup dataset than that of the IMDb dataset (0.8330 and 0.8399). These results imply that accuracy for binary classification (IMDb) has a smaller correlation with training dataset size than that of multiclass classification, where training dataset size can contribute greatly to prediction accuracy. In an additional experiment, we discovered that introducing n-grams of size ≤ 2 (word-pair features) achieved slightly better test accuracy on the IMDb dataset (Naive Bayes: 0.8729 vs. 0.8393, Logistic Regression: 0.8965 vs. 0.8832) and slightly worse test accuracy on the Newsgroup dataset (Naive Bayes: 0.6463 vs. 0.6686, Logistic Regression: 0.6082 vs. 0.6168).

2 Datasets

2.1 20 Newsgroup

The 11,314 instances of training data were fetched from the default train subset in sklearn, consisting of news text from one of 20 pre-determined categories such as computer graphics, motorcycles, medicine, the Middle East, and hockey. Labels ranging from 0 to 19 were assigned to each of these categories, respectively. Each piece of text was converted to word frequency features using `CountVectorizer` (fitted on the training data) ignoring words occurring fewer than 10 times in the text. This reduces the number of features for each sample from 101,631 to 10,739, significantly simplifying our model. Despite this simplification, we found performance improved with this choice as it removed features representing rare typos and words that don't occur frequently enough to add value to predictions. Furthermore, removing these words also helped to reduce overfitting to rare words. The data was then transformed with `TfidfTransformer` (fitted on the training data) which calculated the word features' inverse document frequency, giving more weight to high information (low frequency) words. The distribution of classes in 20 News Groups is shown in *Figure 1*.

Figure 1: Class Counts for the 20 News Group Dataset



2.2 IMDb Reviews

The training data, which consists of 25,000 highly polar (12,500 positive and 12,500 negative) movie reviews, was extracted from the dataset's `train` folder and loaded into two arrays (the training text reviews and their corresponding labels). Labels of 1 and 0 were assigned to positive and negative reviews, respectively. As with the Newsgroup dataset, each text review was first transformed with `CountVectorizer` (count $\geq$ 10) and then with `TfidfTransformer`. For the IMDb dataset, the (count $\geq$ 10) setting reduced the number of features from 74,849 to 18,523.

3 Results

3.1 Experiment 1: Hyperparameter Search with 5-Fold Cross Validation

Both algorithms underwent hyperparameter search with 5-Fold Cross Validation.

The Naive Bayes algorithm was tested with two hyperparameters: `laplace_val` (1, 2, and 3) and `log_correction` (0.001, 0.01, and 0.1), which prevents features that never occur in a class from messing up log values in the `predict`

function. The highest 5-fold cross-validation accuracy for the IMDb data was 0.8572, with `laplace_val` = 1 and `log_correction` = 0.01. The highest 5-fold cross-validation accuracy for the Newsgroups data was 0.7194, also with `laplace_val` = 1 and `log_correction` = 0.01.

The Logistic Regression algorithm was tested with one hyperparameter: `C` (100, 10, 1, 0.1, 0.01), an inverse regularization parameter inversely proportional to lambda. The highest 5-fold cross-validation accuracy for the IMDb data was 0.8844, with `C` = 10. The highest 5-fold cross-validation accuracy for the Newsgroups data was 0.6672, with `C` = 100. Note: We used the fixed hyperparameter `n_jobs = -1` to tell the model that it can use all of the CPUs.

3.2 Experiment 2: Feature Performance

The optimal hyperparameters obtained from the best 5-fold cross-validation of both models on each dataset were then refit on the full training data. Performance was then reported on the test data and is summarized in *Figure 2*.

Figure 2: Performance Comparison between NB and LR

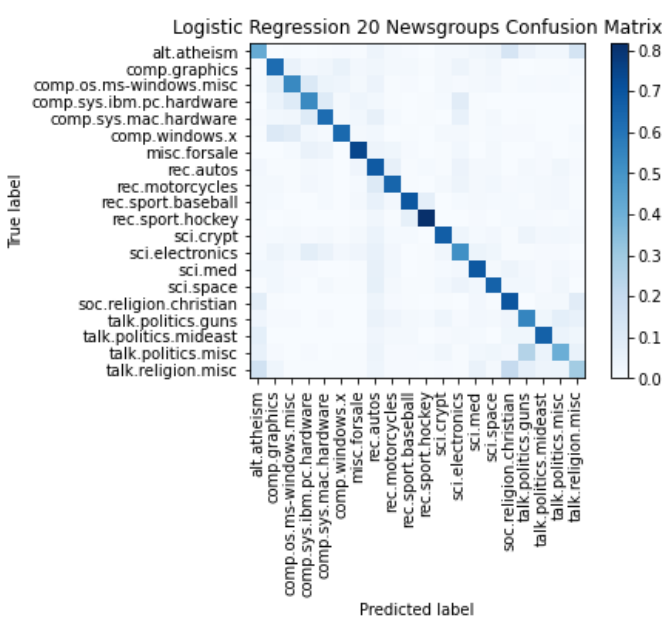
	Accuracy
NB IMDB	0.839320
LR IMDB	0.883200
NB 20 Newsgroups	0.668614
LR 20 Newsgroups	0.616835

For the IMDb test accuracy, the Logistic Regression model outperformed the Naive Bayes model, 0.8832 to 0.8393. For the Newsgroup test accuracy, on the other hand, the Naive Bayes model outperformed the Logistic Regression model, 0.6686 to 0.6168.

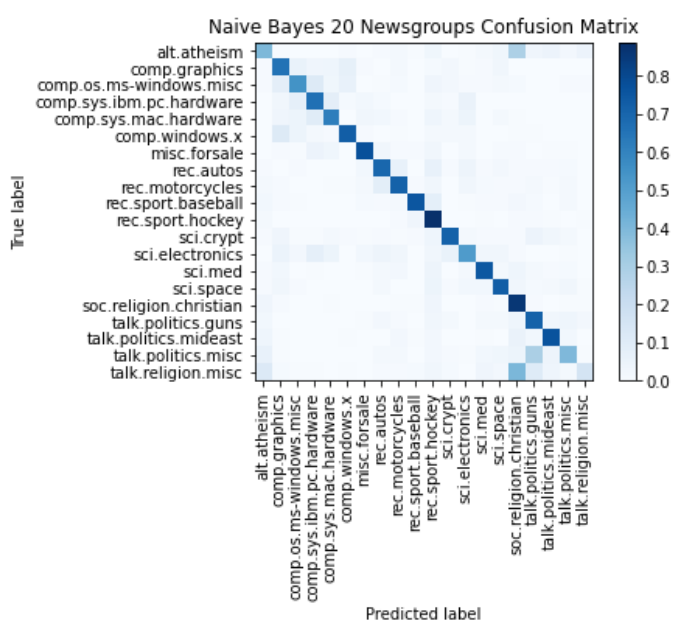
Illustrated in *Figure 3 (ab)*, we also computed confusion matrices for both models on the 20 Newsgroups dataset, which map the rate of successfully classifying the text data by category: predicted labels are charted against the true labels. As you can see from the figures, the Logistic Regression model was slightly better at correctly classifying the text. Furthermore, the classes that are falsely predicted most often are the ones with related topics such as *talk.politics.misc* and *talk.politics.guns* which indicates that even when incorrect, the models are still making reasonable predictions.

Figure 3: Confusion Matrices for 20 News Groups Dataset

(a) Logistic Regression



(b) Naive Bayes

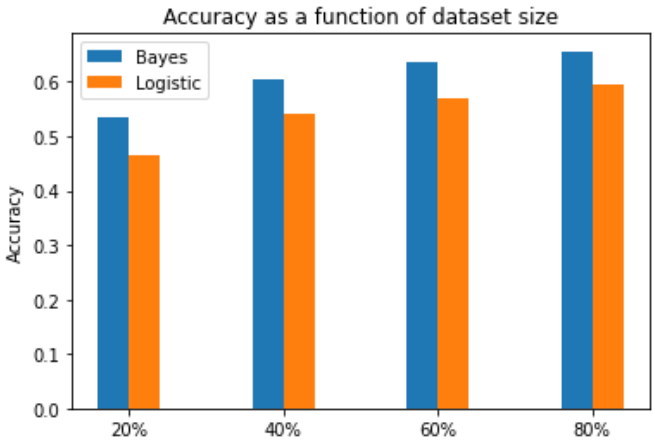


3.3 Experiment 3: Accuracy as a Function of Dataset Size

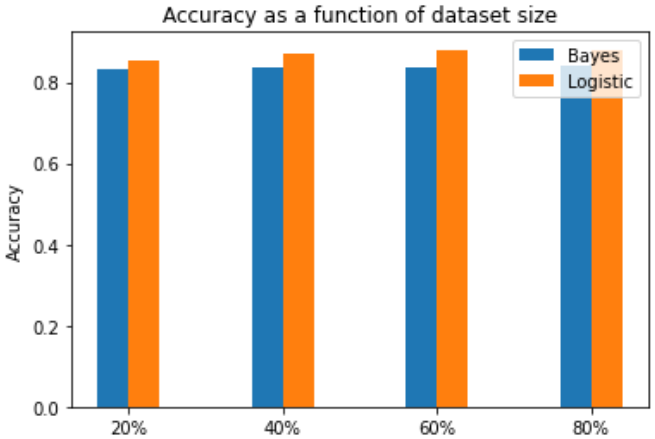
For both datasets, 20%, 40%, 60%, 80% of the available training data were randomly selected to train both models. The hyperparameters were chosen from previous experiments. Then, the models were tested with the original testing data, and their accuracy was evaluated. The results can be seen in *Figure 4*.

Figure 4: Accuracy as a Function of Dataset Sizes

(a) 20 Newsgroups plot



(b) IMDb plot



(c) 20 Newsgroups table

	bayes accuracy	logistic accuracy
20%	0.535847	0.465082
40%	0.605815	0.542087
60%	0.636883	0.570101
80%	0.656267	0.596389

(d) IMDb table

	bayes accuracy	logistic accuracy
20%	0.83300	0.85152
40%	0.83776	0.86876
60%	0.83848	0.87692
80%	0.83992	0.88012

For the 20 Newsgroup dataset, the impact of a reduced training data availability was greater compared to the IMDb dataset. There was a far greater difference between the 20% accuracy and 80% (0.5358 and 0.6563 respectively) dataset availability in the Naive Bayes model. Similar results follow for the Logistic Regression model, where there are large differences between low and high dataset sample size (0.4651 and 0.5964). In contrast, a change in data availability for the training IMDb dataset did not significantly affect the accuracy score for either the Naive Bayes model or the Logistic Regression (0.8330 and 0.83992; 0.85152 and 0.8801). From this, it can be implied that accuracy for binary classification (IMDb) has a smaller correlation to training dataset size compared to the multiclass classification, where training dataset size can contribute greatly to prediction accuracy.

Overall, for the purpose of multiclass classification on the 20 Newsgroup dataset, the Naive Bayes model performed better consistently in regards to prediction accuracy. In contrast, for the binary classification of the IMDb dataset, the Logistic Regression model performed better regardless of the size of the dataset provided.

3.4 Experiment 4: N-grams (Extra)

Our models for Experiments 1-3 relied exclusively on word frequency data, the shortcomings of which are described in detail in *Section 4: Discussion and Conclusion*. Thus, we incorporated n-grams into the `CountVectorizer`, which would count occurrences of a sequence of n-words together. For example, n = 2 counts pairs of words that occur. This might allow the model to capture something like "not good" as a meaningful pair of words and thus have more predictive power in discriminating between text categories.

Using the optimal hyperparameters obtained in Experiment 1 and n-grams of size ≤ 2, the new models had slightly better test accuracy on the IMDb dataset (Naive Bayes: 0.8729 vs. 0.8393, Logistic Regression: 0.8965 vs. 0.8832). This may be explained by the fact that many positive or negative reviews are determined by simple negations, giving more predictive power to pairing words (e.g., "not good"). However, the new models had slightly worse test accuracy on the Newsgroup dataset (Naive Bayes: 0.6463 vs. 0.6686, Logistic Regression: 0.6082 vs. 0.6168). This may be explained by the fact that having word pairs won't necessarily improve category matching in this case because the words themselves are topic-related.

4 Discussion and Conclusion

While our results in Experiments 1-3 were not too poor, because our models in Experiments 1-3 relied solely on word frequency data for their predictions, the context of the words - and other information besides word frequency - was lost. This was especially relevant for the IMDb positive vs. negative reviews: one could theoretically have sentences such as "I do not *like* this movie" where the model detects the word *like* and designates the statement as a positive review/prediction without understanding the negation contextually. As the model relies on word frequency, the order of words (which would then give different meanings to the sentence) is lost as well: "I do not love this movie. In fact I hate it" and "I do not hate this movie. In fact I love it" would have the exact same feature vectors when passed into our models. Other forms of expression, such as sarcasm, are also well beyond the model's comprehension.

While our addition of n-grams was a step in the right direction, possible directions for future investigation would entail more powerful machine learning models like Recurrent Neural Networks, which take, for example, word positioning into account.

5 Statement of Contributions

Contributions by Schmitt-Ulms: Experiment 1, 2 and parts of 3 coding, Report Editing
Contributions by Hong: Experiment 3, LaTex Figures, Report Writing
Contributions by Korsunsky: Experiment 1, 2, Report writing.

References

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

[2] Maas, A., Daly, R., Pham, P., Huang, D., Ng, A., & Potts, C. (2011). Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (pp. 142–150). Association for Computational Linguistics.

[3] A. O. Adi and E. Çelebi, "Classification of 20 News Group with Naïve Bayes Classifier," 2014, *22nd Signal Processing and Communications Applications Conference (SIU)*, Trabzon, Turkey, 2014, pp. 2150-2153