Naive Bayes and Logistic Regression for Text Classification

Saumyaa Verma, Alex Wang, Julia Sokolov

ABSTRACT

In our project we conduct Natural Language Processing tasks on two data sets - 20NewsGroup Dataset[1] and Sentiment140 Dataset[2]. We implement the Logistic Regression model from the Scikit Learn Library and Naive Bayes from scratch. We conduct multiple preprocessing techniques to achieve a higher accuracy. We also conduct hyper-parameter search along with K-fold cross-validation to find the best hyper-parameters for our models used in these text classification tasks. Overall, for the Sentiment140 Dataset, with the best hyperparameters and entire training set, Naïve Bayes outperformed Logistic Regression on unseen test data. While for the 20 News Group Dataset, Logistic Regression outperforms Naïve Bayes. In general, the Sentiment140 Dataset outperforms the 20Newsgroup Dataset in testing accuracies. For both the datasets, Multinomial Naïve Bayes ran faster to train than the Logistic Regression model.

1. INTRODUCTION

In this report we describe the steps we took to implement the Naive Bayes algorithm and Logistic Regression along with best hyper-parameter search, over two textual data sets: one for news data and one for twitter sentiment data. We wanted to compare the effectiveness of both methods in learning and predicting data along with seeing which hyper-parameters gave us the best results.

The 20 Newsgroups dataset contains around 18,000 news articles evenly distributed among 20 distinct categories. 11,000 articles were designated as the training set and the remainder as the test set. Each article has a header with a subject line and signature files, and some contain encoded images and quotes from other articles. The dataset was published in 1999 and the articles were written throughout the 1990s. Each article is a separate file so ample preprocessing is needed before use.

Conducting hyperparameter search on 20 Newsgroups using Logistic Regression yielded a peak test accuracy of 68.8% using the hyperparameters 'C': 10, 'penalty': 'l2', 'solver': 'liblinear'. The training accuracy of this model was 97.1%, and its average cross-validation accuracy was 75%. Using Multinomial Naïve Bayes, the highest test accuracy found was 65.10% when using smoothing parameter, α, as 1.0, and the training accuracy was 81.32%

Many papers have used 20 Newsgroups to develop text classification models such as Spherical K-Means and Neural Variational Inference. A paper about hyperparameter transfer intrigued us most because it describes how we can attribute the optimal hyperparameters to a group of datasets with similar characteristics (Terragni Harrando Lisena Troncy Fersini, 2022) [3]. We could save precious time and computing power by not having to perform cross validation to tune the hyperparameters for each dataset.

The Sentiment140 data set contains textual data that consists of the following columns: the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive), the id of the tweet (1234), the date of the tweet (Sun March 6 23:58:44 UTC 2022), the query type, the user that tweeted (coffee_addict) and the text of the tweet (I love coffee). Here our target/class variable was the polarity of the tweet and classification was done based on the tweet itself. Hence, the other redundant columns (tweet_id, user_id and query_type), were dropped. The training dataset consisted of 1.6 million tweets and the test dataset consisted of 498 data points.

While conducting our analysis, we found that the Sentiment140 Dataset performed the best on the Multinomial Naïve Bayes model with a testing accuracy of 81.61% with the best alpha hyperparameter. With the best hyperparameters on the full training set, it achieved a testing accuracy of 80.2% using the Logistic Regression Model with hyperparameters: 'C': 10, 'penalty': 'l2', 'solver': 'lbfgs', multi_class= "multinomial".

There has been a lot of related work that has been done using the Sentiment140 Dataset. We found the paper on enhancing the performance of sentiment analysis by feature selection and combination (Iqbal Chowdhury Ahsan, 2018) [4] interesting as is suggested multiple methodologies to evaluate effectiveness of a model like recall, precision, accuracy and F1 score. The paper also showed how good feature selection can enhance model performance immensely.

2. DATA

2.1. 20 Newsgroup Dataset

We will first inspect the 20 Newsgroups training set. The distribution of articles among the 20 categories is even, at around 600 articles per category. However, we can condense the 20 categories into 6 broader classes because many categories

such as comp.os.ms-windows.misc and comp.windows.x are remarkably similar. After grouping the categories, we note that the training set is heavily biased toward technology since around half of the articles are classified as either science or computers. Religion was fairly underrepresented. The class distribution and top tfidf words are given below:

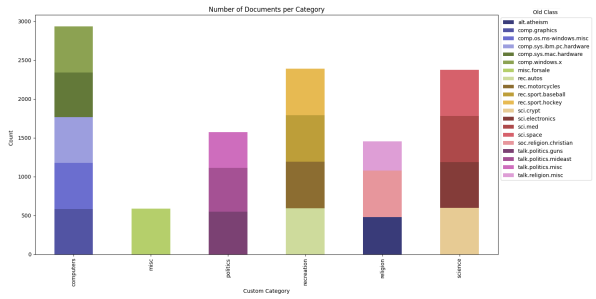

Fig. 1: Class Distribution

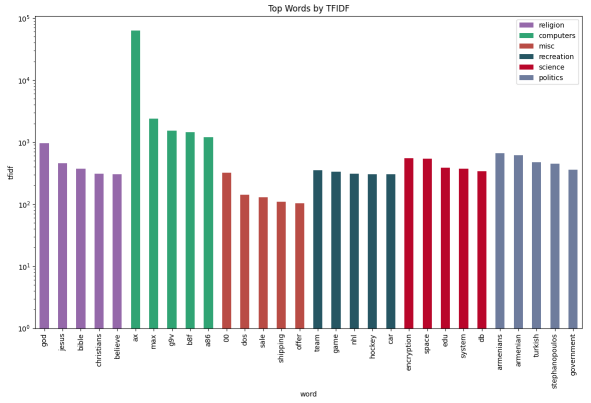

Fig. 2: Top Words by TFIDF

The most striking result was the word "ax", whose TFIDF score in the "computers" category was at least 26 times higher than every other word in all categories. This was the reason why we had to use a logarithmic scale in this plot. Upon further inspection, terms such as "ax" and "g9v" occur primarily in encoded images. The training set, particularly the computers category, happens to contain many documents with encoded images.

The other results are fairly expected. The recreation section is dominated by hockey. The science section is mainly interested in data security and outer space. The political discussion focused on the Turkish-Armenian conflict and George Stephanopoulos, a prominent news anchor and former political advisor.

The test set's distribution among the categories is identical to that of the training set, satisfying the fundamental assumption of machine learning. The test set's raw size is smaller at a total of 7,000 articles instead of 11,000. The top 5 words by TFIDF per category are nearly the same except the computers class, where words associated with image encoding such as "ax" and "g9v" are no longer present.

2.2. Sentiment Dataset

The Sentiment140 dataset was already divided into a train and test set. The train set consisted of 1.6 million data points initially while the test set consisted of 498 data points. The test and the train set consisted of no null values. As for the class distribution, the train set consisted of equal numbers of positive and negative tweets (800,000). The test set on the other hand, consisted of tweets with polarity 0, 2 and 4, where 2 signified neutral tweets, which we went ahead and dropped. After dropping the neutral tweets, we ended up having 177 negative and 182 positive tweets for a total of 359 test points. This can be seen in the class distribution plots below:

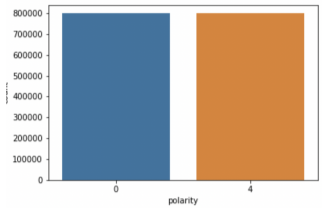

Fig. 3: Train Set

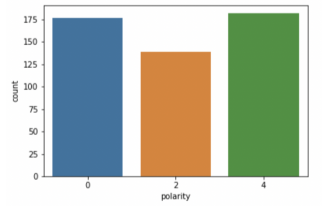

Fig. 4: Test Set (with neutral tweets)

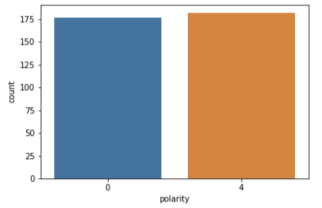

Fig. 5: Test Set (without neutral tweets)

To extract features from the dataset, we conducted multiple steps of pre-processing. First, we equated the polarity of positive tweets to 1 and negative to 0. Then, all the tweets were converted to lowercase. Then all URL's and numbers were removed, however, tags were kept. Contractions were changed to the root word (won't → will not) and punctuation and stop words were then removed. Finally, the words in the tweets were tokenized and lastly, lemmatized (running, ran → run).

Lemmatizing changes the words to their roots and hence leads to an increase in accuracy by extracting the most important features. All these techniques allowed us to have only words of high importance and moreover only the root words. After the cleaning, some tweets became NULL (ex: a tweet with only stop words). These tweets were then dropped, and our final training dataset size was 1592100. To get an idea of which words occurred in positive tweets and negative tweets, we also created word clouds.

Fig. 6: Top 100 words for Negative Tweets

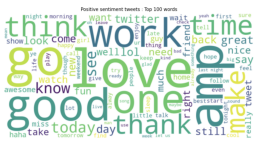

Fig. 7: Top 100 words for Positive Tweets

Then, we applied the Bag of Words approach on our tweets through a CountVectorizer and finally used the TFIDF transformer to extract the top features.

3. RESULTS

3.1. Multiclass classification on the 20NewsGroup Dataset and the Sentiment140 Dataset

We conducted 3 preliminary trials to classify the 20 Newsgroups dataset and progressively boosted the test accuracy each time. The trials and their test accuracies are listed in Table1. Using GridSearchCV and our cross_validation_split

Trial	*Method*	*Accuracy*
Trial 1	Whole dataset using CountVectorizer	60%
Trial 2	Stop words = English, no numbers, punctuation, case insensitive	63%
Trial 3	Using TfidfVectorizer with cleaning from Trial 2	68%

Fig. 8: Table1

function, we concluded that the best combination of hyperparameters was C=10 with a liblinear solver and L2 regularization. The average cross validation accuracy was an impressive 75%. This model had a test accuracy of 68.76%, the highest of any Logistic Regression classifier we used.

Interestingly, the best Logistic Regression model did not use L1 regularization, which performs feature selection. The model that dropped certain features performed worse, suggesting that all features in the training set were significant in predicting the test set.

```
Best Accuracy: 74.97% using {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}
Accuracy 66.45%, sd 1.5% using: {'C': 0.1, 'penalty': 'l2', 'solver': 'newton-cg'}
Accuracy 66.45%, sd 1.5% using: {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
Accuracy 66.42%, sd 1.5% using: {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
Accuracy 35.57%, sd 0.6% using: {'C': 0.1, 'penalty': 'l1', 'solver': 'liblinear'}
Accuracy 73.53%, sd 0.8% using: {'C': 1, 'penalty': 'l2', 'solver': 'newton-cg'}
Accuracy 73.53%, sd 0.8% using: {'C': 1, 'penalty': 'l2', 'solver': 'lbfgs'}
Accuracy 73.37%, sd 0.9% using: {'C': 1, 'penalty': 'l2', 'solver': 'liblinear'}
Accuracy 64.11%, sd 0.5% using: {'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}
Accuracy 74.84%, sd 0.7% using: {'C': 10, 'penalty': 'l2', 'solver': 'newton-cg'}
Accuracy 74.85%, sd 0.7% using: {'C': 10, 'penalty': 'l2', 'solver': 'lbfgs'}
Accuracy 74.97%, sd 0.8% using: {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}
Accuracy 69.92%, sd 1.0% using: {'C': 10, 'penalty': 'l1', 'solver': 'liblinear'}
```

Fig. 9: Grid search for the Best Hyperparameters

When implementing Naïve Bayes on the 20 News Group dataset we chose to use the CountVectorizer function to receive discrete data and worked with both binary results and word frequencies. We initially performed Categorical Naïve Bayes and used bag of words representation, setting each instance for each value equal to 0 or 1, but the accuracies were fairly low. We then chose to implement Multinomial Naïve Bayes, this time using word frequencies, and received better accuracies along with a much faster speed.

When using Multinomial Naïve Bayes on the 20 News Group dataset, we used various subsets in an attempt to improve accuracy. The subsets and their accuracies are shown in the table2:

Trial	*Subset*	*Accuracy*
Trial 1	Whole dataset	54.31%
Trial 2	Stop words = English and stripped accents	63.44%
Trial 3	Stop words = English, stripped accents, and set min_df = 3	65.10%
Trial 4	Stop words = English, stripped numbers, punctuation, and non-ASCII characters	63.99%
Trial 5	Stop words = English, stripped numbers, punctuation, and non-ASCII characters, and set min_df = 3	64.98%

Fig. 10: Table2

As seen in the table above, we were able to increase accuracy from 54% to 65%. We then used the trial with the highest accuracy, Trial 3, to perform hyperparameter tuning. For Naïve Bayes, we chose to tune the hyperparameter α. We then performed 5-fold Cross-Validation and found that the best value of alpha is 1.0.

On the Sentiment140 dataset, we conducted multiclass classification by using CountVectorizer (Naive Bayes with discrete data) or along with TFIDF transformer (Logistic Regression) on a) The original dataset and preprocessing but no lemmatization, removal of stop words or tokenization and b) the dataset with all the preprocessing (including lowercasing, removal of URL's and Numbers, expansion of contractions, removal of punctuations and stop words, tokenization and lemmatization). For Multinomial Naïve Bayes with the default attributes, case a) achieved a testing accuracy almost 2 percent lower than case b). Removal of these noisy elements and hence feature reduction, hence led to an increase in testing accuracy in the case of Multinomial Naïve Bayes. The training accuracy in case a) was 78.96% and in b) was 76.71% which means that our feature selection methodology

has also reduced overfitting.

In terms of logistic regression, we conducted hyperparameter tuning by using grid search and our custom cross validation. Using this, we were able to identify the best hyperparameters: 'C': 10, 'penalty': 'l2', 'solver': 'lbfgs', multi_class= "multinomial". With these, we were able to achieve a testing accuracy of 80.2% and a training accuracy of 77.4%. Since this was done using the preprocessed dataset with feature reduction already done, it is understandable that L2 performed better than L1 as the left-over features were all significant. In this model, for feature selection, we observed that using just CountVectorizer or TfidfVectorizer gave lower accuracies by around 1.5% than if we used CountVectorizer and then TFIDF transformer. Hence, we utilized the latter in our models.

For Naïve Bayes, we conducted hyperparameter tuning by using our custom KfoldCV and cross validation. Using this, we were able to identify the best hyperparameter "alpha" (smoothening constant): 0.7. With this alpha, we were able to achieve a testing accuracy of 81.6% and a training accuracy of 76.7%. This was done using the CountVectorizer as we used Multinomial Naïve Bayes. We also experimented using a Categorical Naïve Bayes model, however since the Multinomial one performed marginally better, we went ahead with it. Since Multinomial Naïve Bayes utilizes discrete data, we used CountVectorizer for our feature extraction. The variation of validation accuracy with alpha is given below:

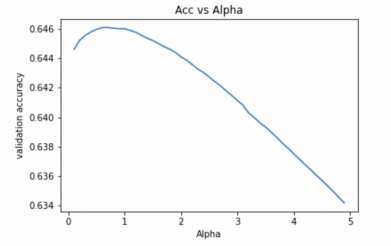

Fig. 11: Sentiment140

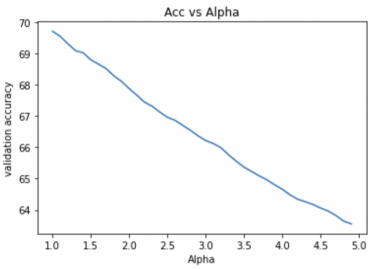

Fig. 12: 20Newsgroup

Model	Dataset	Test Accuracy	Training Accuracy	Hyperparameters
Naïve Bayes	140Sentiment	81.6%	76.7%	α = 0.7
Logistic Regression	140Sentiment	80.2%	77.4%	'C': 10, 'penalty': 'l2', 'solver':'lbfgs"'
Naïve Bayes	20 Newsgroups	65.10%	81.32%	α = 1.0
Logistic Regression	20 Newsgroups	68.8%	97.1%	'C': 10, 'penalty': 'l2','solver':'liblinear'

Fig. 13: Table3

3.2. Comparison between Naïve Bayes and Softmax Regression (table3)

For the Sentiment140 dataset, Naïve Bayes is the winner. It performs better than the Logistic Regression model on unseen data and a lower training accuracy signifies lower overfitting. The accuracies are all very close though so Naïve Bayes performed marginally better.

Conversely, Logistic Regression is the winner for the 20 Newsgroups dataset. Its test accuracy is higher by a significant 3%; however, it's extremely high training accuracy may suggest overfitting.

Both algorithms performed significantly worse on 20 Newsgroups, likely because it contained meaningless words such as "ax", "g9v", and "giz" that are associated with encoded images and archives. In contrast, 140 Sentiments becomes mostly plain English after symbols such as punctuation and emojis are stripped. Moreover, the effort required to achieve high accuracy on a dataset with 20 categories far exceeds that on a dataset with 2 categories simply because the probability of classifying correctly by chance is much lower.

Overall, the Naïve Bayes outperformed Logistic regression in testing accuracy.

3.3. Accuracy of the two models as a function of the size of dataset

As a final step to tune our data, we ran our models on subsets of the entire training set. We experimented with subsets of 20%, 40%, 60%, 80% and the entire training set. In both the datasets increasing the size of the training set increases the testing accuracy in general however, there a drop in the case of the Naïve Bayes model for Sentiment140. Since our training sets were chosen at random, this can be justified by saying that the 80% dataset had data points that were highly representative of the test set. For Sentiment140, Naïve Bayes performed better than Logistic regression on all train test splits, whereas it was the opposite for the 20Newsgroup Dataset. These results are shown in figures 14 and 15.

4. DISCUSSION AND CONCLUSION

4.1. Key Takeaways

Both datasets require ample preprocessing before use. They contain mainly textual features so these must be converted to numerical features before any ML algorithm can execute. Furthermore, they contain a lot of noise such as symbols,

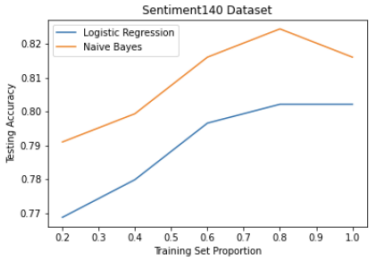

Fig. 14: Sentiment140

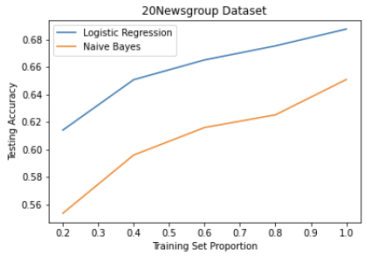

Fig. 15: 20Newsgroup

emojis, punctuation, and grammatical words that must be removed if we hope to achieve a classification accuracy higher than 50

As a general takeaway, with the best hyperparameters, Naïve Bayes was the best model for unseen test data. However, Logistic Regression with it's best hyperparameters outperformed Naïve Bayes on the 20Newsgroup Dataset. Also, for the Sentiment dataset we observed that both Categorical and Multinomial Naïve Bayes had similar results, yet interestingly for the 20 News dataset, Multinomial Naïve Bayes significantly outperformed Categorical Naïve Bayes.

4.2. Possible directions for future investigation

In the future, we would like to clean our datasets even more. Currently, we only conduct lemmatization. However, it would be interesting to conduct stemming and other feature extraction techniques. We would also try to experiment with other models like LinearSVC to extract good features from our datasets.

For Naïve Bayes, possible ways to improve performance could be to perform cross-validation using both α and trying various Naïve Bayes models. In our project, we evaluated Categorical and Multinomial Naïve Bayes, but additional models could also be used.

5. STATEMENTS OF CONTRIBUTION

Alex Wang: Analysis of 20 Newsgroups, Logistic Regression on 20 Newsgroups

Julia Sokolov: Naïve Bayes on both datasets

Saumyaa Verma: Analysis of 140 Sentiments, Logistic Regression on 140 Sentiments

6. REFERENCES

[1] *20 newsgroups*. Home Page for 20 Newsgroups Data Set. (n.d.). Retrieved March 7, 2022, from http://qwone.com/ jason/20Newsgroups/

[2] *For academics - sentiment140 - a Twitter sentiment analysis tool*. Sentiment140. (n.d.). Retrieved March 7, 2022, from http://help.sentiment140.com/for-students

[3] Terragni, S., Harrando, I., Lisena, P., Troncy, R., Fersini, E. (2022, February 15). *One configuration to rule them all? towards hyperparameter transfer in topic models using multi-objective bayesian optimization*. arXiv.org. Retrieved March 7, 2022, from https://arxiv.org/abs/2202.07631v1

[4] Iqbal, N., Chowdhury, A. M., Ahsan, T. (2018). Enhancing the performance of sentiment analysis by using different feature combinations. *2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2)*. Retrieved March 7, 2022, from https://doi.org/10.1109/ic4me2.2018.8465673