COMP551 Mini Project2: Naive Bayes and Softmax Logistic Regression

Group 98

Yao Chen (260910483)
Chuyang Zhang (260840707)
Hanzhi Zhang (260908416)

Abstract

In this mini-project, we investigate the performance of the Naive Bayes model and logistic regression model on two different textual datasets: the newsgroup content – *20 newsgroups* dataset and the Twitter tweet – *Sentiment140* dataset. We first import the dataset and preprocess them using the regular expression operations (*re*) package to remove links, numbers, and special characters. Then we use *CountVectorizer* or *TfidfVectorizer* to change the document from the text to the sparse matrix that contains only numbers for classification. Using hyperparameters such as stopwords, maximum features, minimum document frequency, and maximum document frequency can help reduce the number of features and improve the result accuracy. We implemented the Gaussian Naive Bayes (NB) and Multinomial Naive Bayes from scratch and imported the Logistic Regression model from the *sklearn* library. The Gaussian Naive Bayes was significantly slower than the Logistic (softmax) Regression and Multinomial Naive Bayes. Then we perform the N-fold cross-validation for hyperparameter tuning. As a result, we find that Multinomial likelihood has a better performance in the 20 newsgroups dataset (65.4% accuracy), whereas the logistic regression is more accurate in the Sentiment140 dataset (75.4% accuracy). We find that the Naive Bayes algorithm produced a similar accuracy compare to the Logistic Regression algorithm in both datasets with about 2% difference.

Introduction

Text can be a rich source of information. Naturally, our human brain can process and understand various textual content easily. And we are also able to classify the information behind those textual files by just reading through them. However, it is not realistic to manually process a large amount of text in a short time. Nowadays, machine learning is increasingly used to assist with classification, where textual classification is a fundamental machine learning strategy that assigns a specific label to the text so that the text is organized, categorized, and ready for further process.

This mini-project investigates the improvement of the Naive Bayes model and logistic regression model on classifying textual datasets. Newsgroups(*20 newsgroups* dataset) and Twitter tweets(*Sentiment140* dataset) are selected as the source of the textual dataset since both of them are well-known and commonly used in social communication media [1][2]. For the *20 newsgroups* dataset, we classify the text email body into 20 different categories (multi-class). And for the *Sentiment140* dataset, we attempt to predict the polarity of the tweet (binary class). Next, some basic cleaning is applied on both datasets, with some redundant information removed (i.e. the header in *20 newsgroups* dataset, web page link in *Sentiment140* dataset, etc.). After that, we extracted features from text data using Stopword removal, CountVectorizer and Tfidf transformer. Besides, we aim to optimize the prediction accuracy. We tried Gaussian and Multinomial Naive Bayes algorithms and tuned alpha values. We pick several hyperparameters of the softmax regression algorithm to see how much they will affect the accuracy by starting with some empirical tests. Then we make improvements on those variables via 5-fold cross-validation.

As a result, we find that Multinomial Naive Bayes had a way better performance (both running time and prediction accuracy) than Gaussian Naive Bayes in both datasets. We find that Multinomial Naive Bayes had a better performance on *20 Newsgroup* datasets with an accuracy of 65.4%, whereas Softmax regression had higher accuracy on *Sentiment140* datasets with an accuracy of 75.4%.

Datasets

There are two data sets involved in our experiments: 1) *20 newsgroup dataset*: it is imported from the sklearn datasets, which contain 11314 newsgroups posts (instances) that relate to 20 special topics (classes) in the training data and 7532 newsgroups posts in the test data. 2) *Sentiment140* dataset: the training set contains 1600000 tweets, and the test set contains 359 tweets. The training and test sets have six fields: 'polarity', 'id', 'date', 'query', 'user', 'text', respectively. The *Pandas dataframe* and *re* packages are used to manipulate the dataset.

The *20 newsgroup dataset* is imported from the *sklearn.datasets* with the header, footer and quotes removed and only the main text left. Then, the first dataset is pre-processed by removing all the links, numbers and special characters from the data. This is because the links, punctuations, and numbers could be present in any topic, which doesn't give any useful information for classification and reduces classification accuracy. By doing this preprocessing, the number of features is reduced from 101631 to 62794. Then we use the *CountVectorizer* to convert words to the sparse matrix containing numbers corresponding to the specific word in each data. The *CountVectorizer* could also be set with many hyperparameters to reduce further the number of features for the Multinomial Naive Bayes method. The TfidfVectorizer, compared to CountVectorizer, converts words to the sparse matrix that contains frequencies which is for the Gaussian Naive Bayes implementation.

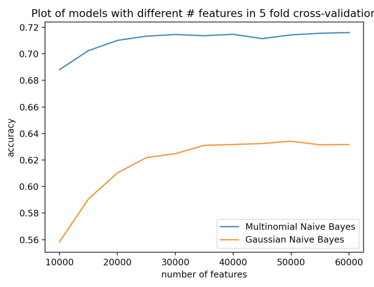
There are six fields in the *Sentiment140* dataset, but we are only interested in the first and the last fields, the 'polarity' and the 'text' of the tweets for the classification. Since the training set has two classes (the polarity of tweets is either positive = 4 or negative = 0) and the test set has three classes (positive = 4, neutral=2, negative = 0), we remove the neutral sentiment using the *loc* method in pandas. Similar to the first dataset, the website URLs and tags are removed first, then the words with special characters are deleted. After that, we removed the rows with duplicate text. At last, the polarity changed from 4, 0 to 1, 0 for easier classification later. The second dataset is also changed to a sparse matrix using CountVectorizer with some hyperparameters to reduce the dimension and get more useful words for classification just the same as the first dataset.

To gain insight into these two datasets, we generated the distribution of the datasets to see if there is any skewness. Referring to Figure A-1 in Appendix Section, it is clear that the *20 Newsgroup* dataset is multi-classed, whereas the *Sentiment140* dataset is a binary classed dataset. Also, we could see that the train-test data has similar distribution and are almost evenly distributed. Therefore, we expect our prediction not to be biased towards any specific class.

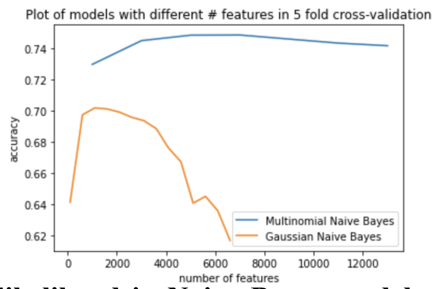
Results

Part1. Naive Bayes Model comparison

After data preprocessing, we start our experiment by testing how different likelihood functions performed on our prediction. We specifically pick Gaussian and Multinomial likelihood since theoretically, the former works well with continuous values where the latter is useful to model discrete data. Running 5-fold cross-validation on both datasets by difference likelihood, refer to the plots below, overall it shows that Multinomial Naive Bayes has higher accuracy than Gaussian Naive Bayes. And exist a parallel trend in Figure 1A, with no such trend in Figure 1B.



A)



B)

Figure 1: Comparison of accuracy of using different likelihood in Naive Bayes model. A) on *20 Newsgroup* dataset and B) on *Sentiment 140* dataset.

Part2. Naive Bayes vs. Softmax regression on different parameter/hyperparameters

To find the highest accuracy of these three algorithms on both datasets, we used 5-fold cross-validation on many hyperparameters.

I. The number of features on both datasets

We first tested how the number of features could affect the accuracy of both datasets. The data is shown below in Figure 2.

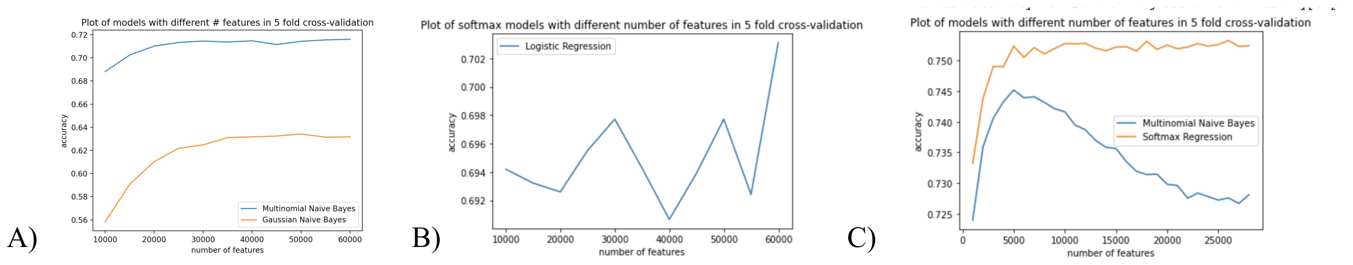


Figure 2: The number of features that affect the prediction accuracy. A) Using two Naive Bayes algorithms on *20 newsgroup* datasets, B) using softmax regression algorithm *20 newsgroup* dataset and C) Multinomial Naive Bayes algorithm and Softmax regression algorithm on *Sentiment140* dataset.

Figures 2A concluded that as the number of features increased, the accuracy increased initially, then levelled off. As for the softmax regression, there is a increasing trend with more features involved. The Multinomial Naive Bayes had the highest accuracy at 60000 features, and the Gaussian Naive Bayes had the highest accuracy at 50000. However, we still set the Gaussian Naive Bayes features to be 10000 to save the running time. The logistic regression model had its highest accuracy at 60000 features as well. While on the *Sentiment140* dataset, picking the Multinomial Naive Bayes model (which had better accuracy base on Part1), it is interesting to see that prediction accuracy reaches a maximum (about 74.5% accuracy) at 5000 features and then decreases with an increasing number of features used. Yet running the softmax regression algorithm, we can see that the accuracy trends converge to around 75.5% as the number of features increases.

II. Experiment on max_df, min_df, and alpha values on both datasets for Naive Bayes model

We then tested how max_df, min_df, and alpha value affect the accuracy for the two Naive Bayes algorithms; the plots are shown below in Figure 3.

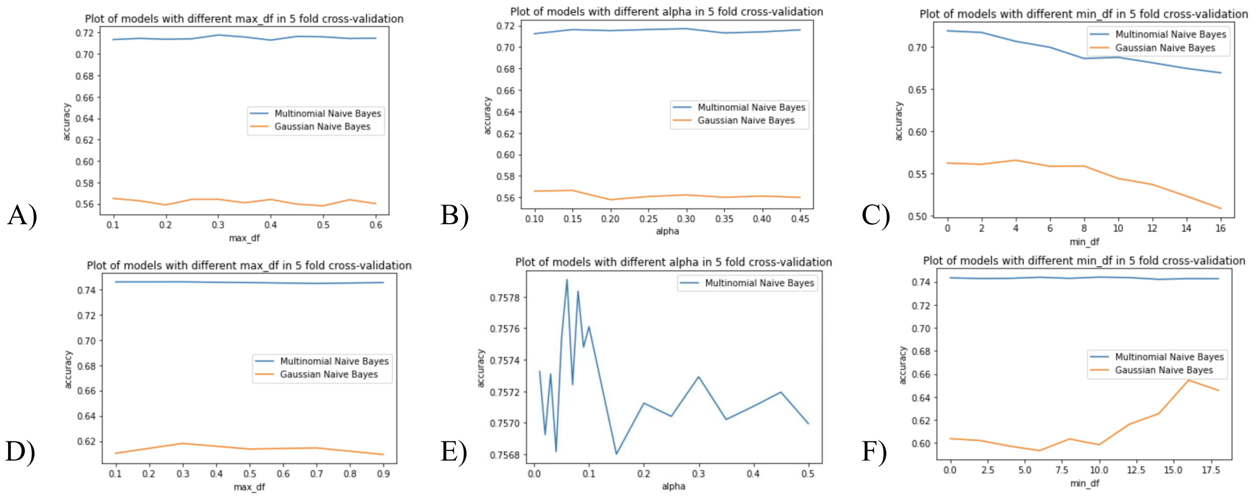
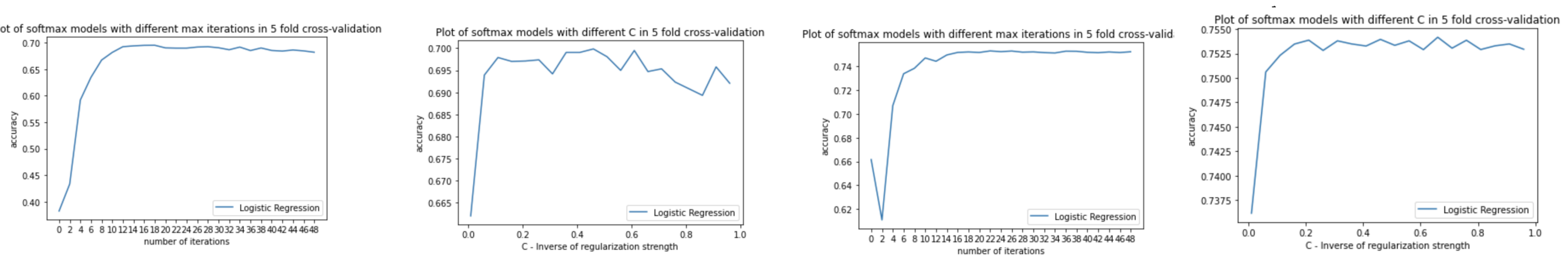


Figure 3: max_df, min_df, and alpha affect the accuracy of the Multinomial and Gaussian NB. A),B) and C) on the *20 newsgroup* dataset, D), E), and F) on the *Sentiment 140* dataset.

Based on Figure 3, we noticed that most plots between two algorithms are parallel, and these three hyperparameters barely affect the accuracy of the prediction. Then by viewing the maxima, we set the max_df to 0.3, min_df to 1 for both naive Bayes algorithms, and alpha value was set to 0.3 for Multinomial and 0.15 for Gaussian to calculate the *20 newgroups* test set's accuracy. And for *Sentiment 140*, we select max_df to be 0.5, min_df be 5, and alpha to be 0.06.

III. Iterations and Inverse of regularization strength for Logistic Regression

We also explored how to optimize the Logistic Regression model. We used 5-fold cross-validation on two other hyperparameters in the Logistic Regression algorithm: the number of iterations and the inverse of the regularization strength. The result is shown below in Figure 4.



A) B) C) D)

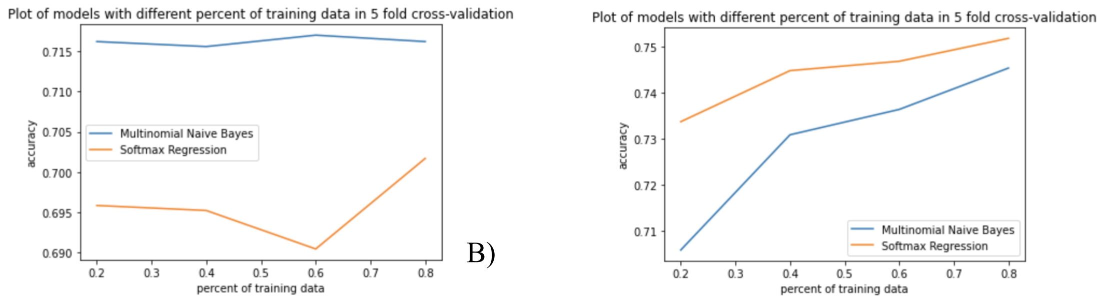
Figure 4: The hyperparameters max_iter and C (inverse of regulation strength). A) and B) the polt of the *20 newsgroup* dataset, C) and D) the plot of the *Sentiment 140* dataset.

Referring to both plots of the number of iterations, we observe that with the increasing number of iterations. It converges to 70% (the *20 newsgroup* dataset) and 75% (the *Sentiment 140* dataset) smoothly. Then the plot of inverse of regularization strength has similar trends, yet the accuracy is not stable. And something to be noticed is that the accuracy generously decreased with the number of iteration/inverse of regularization strength continuously increasing for the *20 newsgroup* dataset. Also, we found that for the *20newsgroup* dataset, the number of iterations and the inverse of regularization strength that produced the highest accuracies were 18 and 0.61, respectively. And for the other dataset, we pick iteration to be 20 and inverse of regularization strength be 0.21.

We lastly compared the penalty between no penalty and ‘l2’ norm penalty in Logistic regression. We found that the accuracy then there’s no penalty is 69.5%, and the accuracy when using the ‘l2’ penalty is 69.9% for the first dataset. The accuracies are 75.1% and 75.4% for the second dataset, respectively.

Part3. Performance of Bayes and Softmax regression on different training set sizes

Furthermore, we modified the size of the dataset to analyze how it affects the accuracy of Bayes and softmax regression algorithms. The figures below represent those algorithms' performance on 20% - 80% of the datasets.



A) B)

Figure 5: Different percentage of a subset of dataset vs. The accuracy of two classification algorithms. A) on *20 Newsgroup* dataset and B) on *Sentiment 140* dataset.

For the *20Newsgroup* dataset, it appears that the changing of size of the dataset would not significantly affect the prediction. In contrast, we could conclude, for the Sentiment140 dataset, with an increasing percentage of training data, the accuracy increases for both algorithms (Multinomial NB (<71% to 74.3%)

and Softmax Regression (73.4% to 75%)). To be noticed, due to the limitation of RAM, we set the training set features to be 100000.

Datasets	Multinomial Naive Bayes	Softmax Regression
20 Newsgroup Dataset	65.4%	63.7%
Sediment180 Dataset	74.7%	75.4%

Table 1: The accuracy of Multinomial NB and Softmax Regression with their optimized parameters/ hyperparameters.

Finally, we summarize the tuned hyperparameters and run the prediction on the test sets, see Table1 and Appendix Table1 for details. The Multinomial Naive Bayes had better performance on the *20 Newsgroup* dataset with an accuracy of 65.4%. And for the *Sentiment 140* dataset, Softmax Regression had a higher accuracy of 75.4%. Also, it is interesting to notice that both algorithms had a better performance on the *Sentiment 140* dataset.

Discussion and Conclusion

In this mini-project, we had a fresh first-handed experience on raw data processing, the Naive Bayes implementation, and the Softmax Regression modelling as well as model optimization.

After that, the overall accuracy improved by the optimized hyperparameters via 5-cross validation (max_df, min_df, and alpha for NB and max_iter and C (inverse of regulation strength) for softmax regression). After Running the optimized algorithms on the final test set, for the *Sentiment 140* dataset, an accuracy of around 75% is achieved. In comparison, the *20 Newsgroup* dataset had 65.4% and 63.7% accuracy on Multinomial NB and Softmax regression, respectively. As a result, both algorithms generally performed better on the *Sentiment 140* dataset compared to the 20 *Newsgroup* set. Another observation is that the test accuracy of the *20 Newsgroup* dataset is lower than training accuracy. After our discussion, we find this issue may be due to the size of the test set. Referring to Appendix Figure A-1, we could see that the total size of the training set is not too small. However, only hundreds of data could fall into each class since there are twenty classes. Since we are using the default splitting, the size of the testing set is almost 38% of the entire data set. Combining the small size of the training sample and relatively large testing set, it is not intuitive to have an overfit in our training process, therefore lowering testing accuracy.

During implementing the Naive Bayes models, we found that the prediction accuracy significantly dropped if a word has 0 appearances in a certain class (since its mean and standard deviation would become 0). To solve this problem, we introduced a hyperparameter, alpha, for calculating the likelihood for Multinomial Naive Bayes. For Gaussian Naive Bayes, we also added a threshold when calculating the log_likelihood to avoid this problem. The threshold was set to constant 1e-20.

To wrap up, we had explored two textual classification algorithms in this project. However, due to the sizes and the numbers of the datasets and the RAM limitation, we can not directly conclude which model is the best for textual classification. Many more advanced classification algorithms such as Linear Support Vector Machine may perform better [3].

From the takeaway of the last mini-project, we improved our coding style, which made the experiment execution easier and more efficient.

Statements of contribution

The group members distributed the workload equally.
Yao: data preprocessing, graph generation, Gaussian and Multinomial Naive Bayes algorithm implementation, creativity.
Chuyang: data preprocessing, graph generation, data generation, write-up, creativity.
Hanzhi: data preprocessing, write-up, creativity.

References

[1]"sklearn.datasets.fetch_20newsgroups", scikit-learn, 2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_20newsgroups.html. [Accessed: 07- Mar- 2022].
[2]A. Go, R. Bhayani and L. Huang, "For Academics - Sentiment140 - A Twitter Sentiment Analysis Tool", Help.sentiment140.com, 2022. [Online]. Available: http://help.sentiment140.com/for-students. [Accessed: 07- Mar- 2022].
[3]B. Stecanella, "Support Vector Machines (SVM) Algorithm Explained", MonkeyLearn|Blog, 2022. [Online]. Available: https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/. [Accessed: 07- Mar- 2022].
[4]"Text-Analytics-20-Newsgroups-Dataset/1. ProjectCode.ipynb at master · rshah204/Text-Analytics-20-Newsgroups-Dataset", GitHub, 2022. [Online]. Available: https://github.com/rshah204/Text-Analytics-20-Newsgroups-Dataset/blob/master/1.%20ProjectCode.ipynb. [Accessed: 07- Mar- 2022].

Appendix

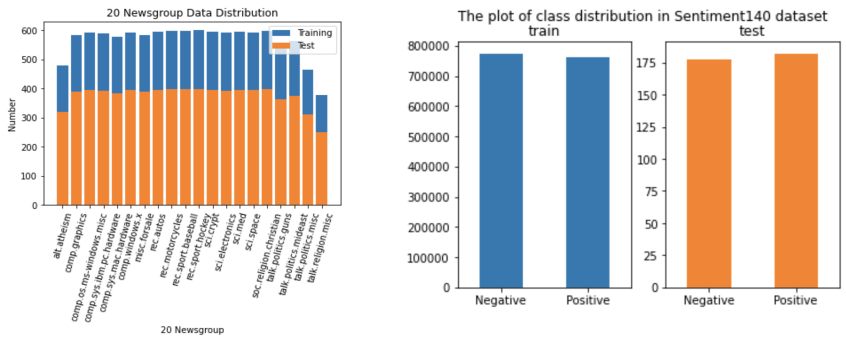


Figure A-1: Distribution of datasets by classes. Left: *20 Newsgroup* dataset and Right: *Sentiment140* dataset.

Table 1: The summary of the performance of Multinomial NB and Softmax Regression with their optimized parameters/hyperparameters.

Datasets		20 Newsgroup Dataset	Sentiment 140 Dataset
Number of features(NB and Softmax regression)		60000, 45000	60000
Multinomial Naive Bayes's hyperparameters	max_df	0.3	0.5
	min_df	2	5
	alpha	0.2	0.06
Softmax Regression's hyperparameters	Number of iterations	18	20
	c	0.61	0.21
Size of training set		all	100000
Accuracy of running models on	Multinomial Naive Bayes	65.4%	74.7%
	Softmax Regression	63.7%	75.4%