# Classification of Textual Data using Naive Bayes and Softmax Regression

COMP 551 Mini Project 2

Chloe Mills, Shania Wan-Bok-Nale, Khoi Nguyen

March 2022

## Abstract

In this project, we were tasked to implement Naïve Bayes classification and Softmax Regression models to classify the following datasets: 20newsgroups and Sentiment140. The 20newsgroups dataset consists of over 18000 newsgroup posts over 20 topics and Sentiment140 dataset consists of a collection of tweets with a positive, negative sentiment. Our goal was to find the model with the highest accuracy. We trained our Gaussian Naïve Bayes model, Multinomial Naïve Bayes model and SciKit learn's Logistic Regression model using different hyper-parameter settings for each of the classifiers, which we found to influence the results. We implemented k-fold cross validation on every model to ensure performance consistency. We found that our best performing model was the Logistic Regression model on 20newsgroup, which reported an accuracy score of 72.28% on the final test set with tuned Hyper-parameters(C=100, Solver = liblinear). On Sentiment140 dataset, our Multinomial Naive Bayes model using tfidf was our best performing model with a score of 76.88% on the final test set. Overall, Softmax Regression was significantly faster to train over both datasets.

## Introduction

Our task was to create a Naïve Bayes Machine Learning model to run on the 20newsgroup dataset and Sentiment140 dataset to classify text into category and sentiment respectively. With 192 million daily active Twitter users and more than 100 million daily tweets, creating a model that could classify text into binary sentiments would be highly desirable. Likewise, there are millions of news group posts each pertaining to different categories. We wanted to create a model that could classify different posts into topics. The ability to classify raw data into categories would allow researchers to organize documents or tweets into specific categories. We trained the following models: Gaussian Naïve Bayes, Multinomial Naïve Bayes and Logistic Regression using a training set of 10,000 randomly shuffled tweets from Sentiment140 and the complete dataset of 20newsgroups with 18,846 newsgroup posts. We decided on the more appropriate model for each dataset based on the test accuracy. The classifiers used were Logistic Regression from SciKit-Learn library, and our own implementation of Multinomial Naive Bayes and Gaussian Naives Bayes. The different text to feature vectorizers used were TF-IDF vectorizer and count vectorizer. Each model was trained with different hyper-parameter settings over K-fold cross-validation to achieve the highest accuracy. After careful experimentation, we found Softmax Regression to be our best classifier model with a test accuracy of 72.28% on the test set of 20newsgroups dataset and Multinomial Naive Bayes model with a test accuracy of 76.88% on the Sentiment140 test set.

## Dataset

The 20newsgroups dataset consists of 20 Classes: alt-atheism, comp-graphics, comp-os-ms-windows-misc, comp-sys-ibm-pc-hardware, comp-windows, comp-sys-mac-hardware, misc-forsale, rec-autos, rec-motorcycles, rec-sport-baseball, rec-sport-hockey, sci-crypt, sci-electronics, sci-med, sci-space, soc-religion-christian, talk-politics-guns, talk-politics-mideast, talk-politics-misc and talk-religion-misc. The dataset is split across 20 different newsgroups. Each class represents a different topic that the newsgroup documents belong to. We collected the training data and test data from SciKit-learn's library datasets.fetch_20newsgroups(). We set the parameter subset=train and subset=test respectively and removed headers, footers and quotes. The training set consists of 11314 rows and 7532 rows in the test set.

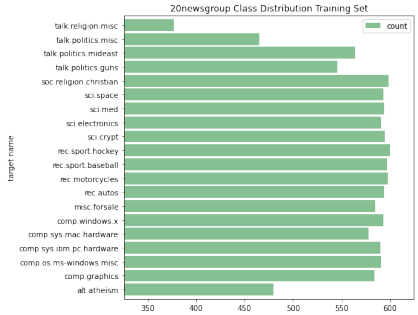


Figure 1: 20newsgroups Training Set distribution over 20 classes

Figure 1 & 2 highlights the similarities between the class distributions for the training set and the test set in the 20newsgroup dataset.

We imported the Sentiment140 dataset into pandas DataFrame and used the text column as the training set and the sentiment column as the target. The Sentiment140 dataset consists 2 classes: Positive and Negative in the training set. The Sentiment140 test set consists of 3 classes: Positive, Neutral and Negative so we removed the neutral class in order to be consistent with training our model. Since the Sentiment140 dataset has over a million tweets, we shuffled

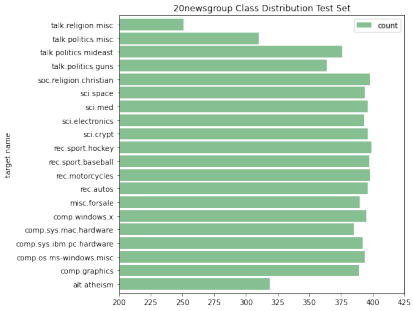


Figure 2: 20newsgroups Test Set distribution over 20 classes

the data with a fixed seed and took the first 10000 shuffled tweets.

In each of the datasets, we extracted the features from the text by counting the occurrence of each word using CountVectorizer.fit_transform() which resulted in a feature vector. We transformed said feature vector into tfidf values by using TfidfTransformer.fit_transform(). In our results we examine whether or not tfidf produces higher accuracy compared to counting occurences. For the target of each respective dataset, we used CountVectorizer() and TfidfTransformer() similarly to the training set. Since the data is already fitted to the training set, we called transform() instead of fit_transform().

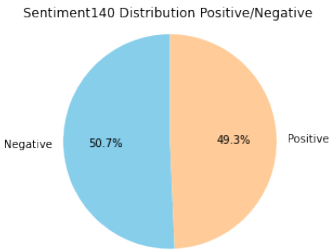


Figure 3: Sentiment140 Test Set distribution over 2 classes

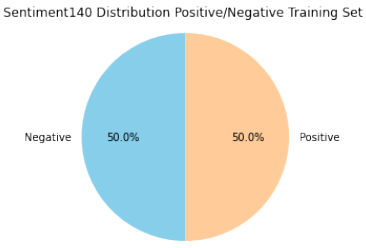


Figure 4: Sentiment140 Training Set distribution over 2 classes

Figure 3 & 4 highlight the similarities between the class distribution for training and test set in the Sentiment140 dataset.

The difference of class imbalance in the training and test sets should be at a minimum in order to ensure the model is trained on similar instances as the test data. Shown in Figure 1,2,3 and 4, we found both datasets to be quite balanced with each class representing a relatively equal weight in each respective dataset. Although a few of the 20newsgroup dataset's classes were on the lower end of the count, we found this to still be balanced.

We further analyzed the datasets by finding the most important word for each of the 20 classes in 20Newsgroups and top 10 most important words of the 2 classes in Sentiment140 using tf-idf. Table 5 in the Appendix highlights our findings on 20Newsgroups Dataset. Table 6 in the Appendix highlights our findings on Sentiment140 Dataset. We find our recent analysis of important words per class to be quite interesting as you can recognize the link between the categories and sentiments to the words at first glance.

## Results

K-Fold Cross Validation: We created the function cross_validation_split to split the data and targets into k=5 folds. Each of the folds is used as validation while the remaining 4 folds are used as the training set. We then implemented the function kfoldCV to perform the 5-fold cross-validation on each of our models, training sets, data and targets. Note that we kept the CountVectorizer() and TfidfTransformer() separate to be able to compare the performance of the K-fold implementation of Naïve Bayes. Gaussian Naïve Bayes: We converted the 20 newsgroup dataset into sparse matrices containing the feature vectors by using CountVectorizer() and TfidfTransformer(). We then converted the feature vectors into numpy ndarrays such that our model could be able to fit and predict the data. This conversion has very high space complexity so we do not save the sparse matrix to ndarray conversion after we are done using it. For example, the fit/predict functions converts the sparse matrix to ndarray, but after the functions are done using it, the space is freed up again. This still caused the predict function to crash when calling it on the entire test set of the 20 newsgroup dataset because this dataset contains over 18000 newsgroup documents. As a result, we split the validation set into subsets by using for-loops and calling each subset sequentially. We saved the results of each subset and calculated the average to obtain the models accuracy. The accuracy of Gaussian Naïve Bayes was relatively low with around 5% on the 20newsgroup dataset since the model predicted that every test case belongs to class 0. The sentiment dataset had an accuracy of around 49% because our model predicted that every test case has negative sentiment (i.e. belongs to class 0). This was a result of fitting the models for each of the datasets with standard deviation equal to 0 for certain features in a class. Hence when calculating the log likelihood used by the posterior in our prediction, the features that have standard deviation = 0 create divide by zero and logarithmic warnings which in turn renders some of the results as NaN. To bypass this error, we added a constant c = 1e-9 when calculating a features standard deviation in our fit function. This increased the accuracy of GaussianNaiveBayes to 55.05% for the 20newsgroup dataset and to 57.38%

for the sentiment dataset.

To achieve higher accuracy, we then implemented MultinomialNaiveBayes classifier. Multinomial naïve bayes performs better than Gaussian naïve bayes because a multinomial distribution can better fit data in the cases that the probability of the features is a multinomial distribution. Counting occurrences works very well under this model. To fit this model, we calculated the priors which is represented by the number of documents in each class, divided by the total number of documents. To perform Laplace smoothing we calculated pi=((Number of occurrences of word in class c)+1)/(total number of occurrences of all words in class c + number of features in c). In the predict function, we calculated the probability that the document belongs to each class and chose the class with the highest probability. We then multiplied the prior of the class by the conditional probability of each word in the document being in said class. Since we are multiplying many small values together, the float gradually becomes smaller until it can no longer represent the number (i.e. underflow). To prevent this we worked in the log domain.

Our first implementation of Multinomial Naïve Bayes took around 2 hours to run for the 20newsgroup dataset since we chose to use for-loops instead of sparse matrices and ndarray functions. In order to optimize the model, we only worked with nonzero elements in the training data which reduced the running time to around one minute. The 20newsgroup dataset had an accuracy of around 52% and the sentiment dataset had an accuracy of around 75%.

Experiments: To obtain the best hyper-parameter for Naïve Bayes we tested our Multinomial Naïve Bayes and Gaussian Naïve Bayes implementation respectively over different text vectorization functions and tested different hyper-parameters. These tests consist of count occurrences with/without stop-words, and tfidf with/without stop-words for each of the datasets.

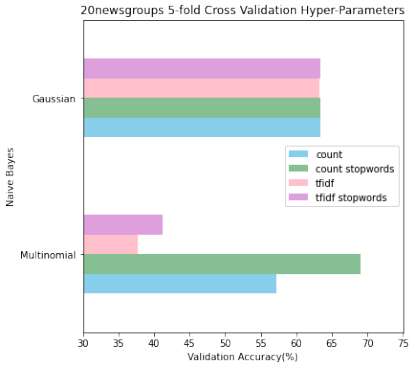


Figure 5: 20Newsgroup Naive Bayes 5-fold classifier comparison

The Sentiment140 dataset performed very well on the 5-fold cross-validation with the MultinomialNaiveBayes model compared to the GaussianNaiveBayes model. When testing different hyper-parameters, the lower bound was around 70% validation accuracy for Multinomial and around 55% for Gaussian as seen in Figure 6.

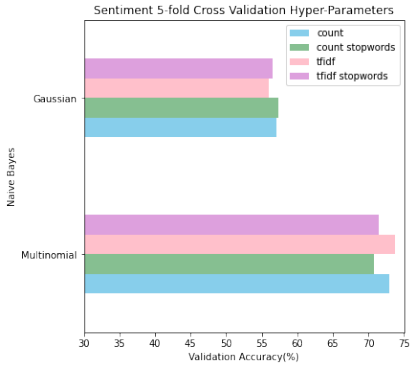


Figure 6: Sentiment Naive Bayes 5-fold classifier comparison

In comparison, the 20Newsgroup dataset had larger variability when testing different hyper-parameters in the 5-fold cross-validation MultinomialNaiveBayes model. Although the highest validation accuracy is a result of the MultinomialNaiveBayes using count with stop-words, the GaussianNaiveBayes performs better on all other hyper-parameters. Figure 5 outlines these results.

Furthermore, we compared our implementation of Multinomial Naïve Bayes to SciKit-learn's with the same hyper-parameters. Our implementation performed slightly better on the 20newsgroups dataset using count occurrences with stop-words than sciKit-learn's implementation using the same hyper-parameters. On the other hand, our implementation performed slightly worse on the sentiment dataset using TFIDF as hyper-parameter compared to SciKit-learns.

Finally, we tested the effect that n-gram have on the test accuracy. The default n-gram is only unigrams (1,1). Since testing unigrams and bigrams increases the space complexity significantly (around 10 times as big), we tested the effect of n-grams on a smaller subset of the training set. This explains the lower test accuracy.

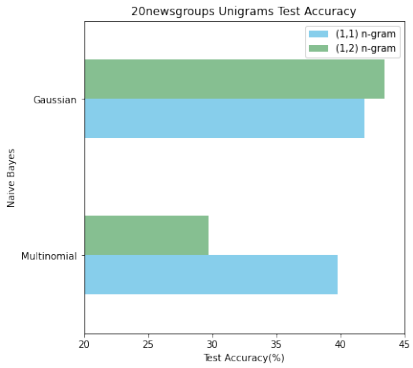


Figure 7: 20Newsgroups Unigrams comparison with best hyper-parameters

Figure 7 & 8 conveys that the Gaussian Naive Bayes model increases in test accuracy when we go from default

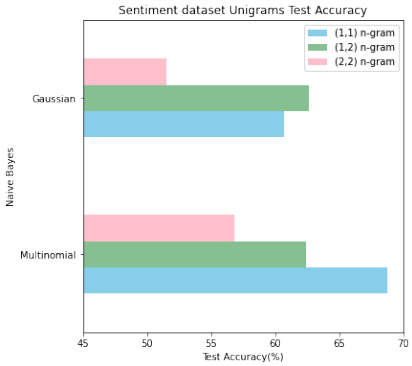


Figure 8: Sentiment Unigrams comparison with best hyper-parameters

(1,1) unigrams to (1,2) both unigrams and bigrams. We assumed that this is because it is equivalent to adding extra training data to the model, therefore leading to an increase in test accuracy. On the other hand, Multinomial Naive Bayes model decreases in test accuracy when we use (1,2) both unigrams and bigrams. We speculate that this is because we have not refined or filtered out important and frequently occuring bigrams in our testing, and were simply using all possible bigrams. This might have had an adverse effect and affected the importance of the unigrams hence causing a decrease in test accuracy.

In table 5 and 6, notice that many of the important features are single words, such as "good", "thanks", "love", "sad". They rarely appear as a bigram. For example, "very good" does not appear as much as "good". Therefore removing unigrams will significantly decrease the accuracy. We have not tested this as extensively on the 20newsgroups dataset compared to the Sentiment140 dataset, however our current results shows a decrease of 42.55% to 29.84% test accuracy on the Gaussian Naive Bayes model with best hyper-parameters, and a decrease of 39.79% to 10.08% test accuracy on the Multinomial Naive Bayes model with best hyper-parameters.

Table 1 highlights our results with the score being the test accuracy. We see that our Multinomial Naive Bayes Model with count and stopwords had the highest test accuracy for the 20 newsgroups dataset while our Multinomial Naive Bayes with tfidf has the highest score for the Sentiment140 dataset.

| Dataset | Best Model/Hyper-Parameters | score(%) |
|---|---|---|
| 20newsgroups | MultinomialNB count & stopwords | 64.76 |
| Sentiment140 | MultinomialNB tfidf | 76.88 |

Table 1: Test accuracy of Naive Bayes over the best Hyper-parameters on both Dataset

Next, we implemented a Softmax regression model using SciKit-learn's LogisticRegression library. When we ran the model with default parameters, we obtained test accuracy of 67.37% on 20newsgroups dataset and 74.09% on Sentiment140 dataset.

We ran Softmax Regression over pairs of different hyper-parameter values on each respective dataset. We chose to run the experiment using the Hyper-parameters C and Solver from the full list of Hyper-parameters given. Table 2 and Table 3 highlight our results with the score being the test accuracy. We see that running Softmax Regression with the hyper-parameters Solver = liblinear and C value = 100, we get the highest test accuracy of 72.28% on 20newsgroup dataset. This is an improvement from the default parameters test accuracy. On the other hand, softmax regression on Sentiment140 with hyper-parameters Solver = newton-cg and C = 1 produced the highest score of 74.00%, which is similar to the default hyper-parameters.

| Hyper-parameter C | Hyper-parameter Solver | score (%) |
|---|---|---|
| 100 | newton-cg | 71.64 |
| 100 | lbfgs | 71.59 |
| 100 | liblinear | 72.28 |
| 10 | newton-cg | 71.49 |
| 10 | lbfgs | 71.50 |
| 10 | liblinear | 72.01 |
| 1 | newton-cg | 69.26 |
| 1 | lbfgs | 69.26 |
| 1 | liblinear | 69.16 |
| 0.1 | newton-cg | 56.53 |
| 0.1 | lbfgs | 56.53 |
| 0.1 | liblinear | 56.51 |
| 0.01 | newton-cg | 42.93 |
| 0.01 | lbfgs | 42.93 |
| 0.01 | liblinear | 44.88 |

Table 2: Test accuracy of SciKit-learn's Softmax Regression over pairs of different hyper-parameters on 20newsgroups dataset

| Hyper-parameter C | Hyper-parameter Solver | score (%) |
|---|---|---|
| 100 | newton-cg | 72.02 |
| 100 | lbfgs | 72.01 |
| 100 | liblinear | 72.02 |
| 10 | newton-cg | 73.39 |
| 10 | lbfgs | 73.38 |
| 10 | liblinear | 73.38 |
| 1 | newton-cg | 74.00 |
| 1 | lbfgs | 73.99 |
| 1 | liblinear | 73.98 |
| 0.1 | newton-cg | 71.40 |
| 0.1 | lbfgs | 71.40 |
| 0.1 | liblinear | 71.44 |
| 0.01 | newton-cg | 66.57 |
| 0.01 | lbfgs | 66.57 |
| 0.01 | liblinear | 67.56 |

Table 3: Test accuracy of SciKit-learn's Softmax Regression over pairs of different Hyper-parameter values on Sentiment140 Dataset

Table 4 displays the model with the hyper-parameters that result in the highest test accuracy for each of the datasets. We can see that the best model for 20newsgroups

| Model | 20newsgroups | Sentiment140 | Mean |
|---|---|---|---|
| MultinomialNB | count & stopwords: 64.76% | tfidf: 76.88% | 70.82 |
| SoftmaxR | C=100, Solver=liblinear: 72.28% | C=1, Solver=newton-cg: 74.00% | 73.14 |
| Best Score(%) | 72.28 | 76.88 | 73.14 |

Table 4: Comparison between Multinomial Naive Bayes and Softmax Regression over the best hyper-parameters.

is Scikit learn's softmax regression with hyper-parameters C=100 and Solver=liblinear. The test accuracy is 72.28%. The best model for Sentiment140 is our implementation of multinomial naive bayes with hyper-parameter tfidf. The test accuracy is 76.88% for Sentiment140 dataset. Overall, the average of the highest test accuracies is 70.82% for multinomial naive bayes and 73.14% for softmax regression. Hence the winner overall is softmax regression.

We ran further experiments with our models by comparing the test accuracy of the two models as a function of the size of training dataset. We selected 20%, 40%, 60% and 80% of the available training data and trained our models on this subset over the two datasets and scored them over their test accuracy. Figure 9 highlights our results. We observe a recurrent trend with the test accuracy increasing as we increase low percentages of the training set to then plateau as we increase high percentages of the training set. At some point we even see that using only 80% of the training set has a slightly better accuracy than using 100% of the training set. We suspect this might arise due to a slight over fit from using 100% of the training set.

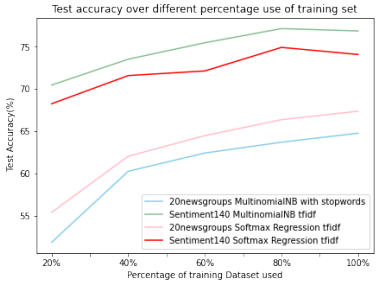


Figure 9: 20newsgroups Test Set distribution over 20 classes

## Discussion and Conclusion

In this project, we classified text into 20 categories and binary sentiments using Naive Bayes and Softmax Regression machine Learning Models over 20newsgroups and Sentiment140 datasets. We examine some points worth discussing.

We observed that our Multinomial Naives Bayes implementation had highly varying test scores when implementing the model with different hyper-parameter functions like tfidf, count, count with stop-words, tfidf, tfidf with stop-words on 20newsgroups dataset. That being said, we found both model to perform quite well. Softmax Regression with hyper-parameters C=100 & Solver=liblinear performed better on the 20newsgroup dataset with a score of 72.28% while our Multinomial Naive Bayes model using tfidf performed better on Sentiment140 with a score of 76.88% on the final test set. As for future investigation, classifying sentiments using a Bernouilli Naive Bayes Machine Learning Model could be a solution as Bernouilli NB works with multiple features but each one is assumed to be a binary-valued variable. We could have also investigated Linear Support Vector Machine model which is regarded as one of the best text classification model.

## Statement of Contributions

All three members of the group contributed equally to this project, we even had a fun time together and it was a great learning curve for all three of us. Nguyen worked on the Multinomial Naive Bayes models while Mills worked on Gaussian naive bayes model. Mills and Wan-Bok-Nale worked on the implementation of Softmax Regression model, analysis and format of the datasets, experiments and results. We all contributed to the write up of the report of our respective work.

## Appendix

| Class | Most important word | tf-idf score |
|---|---|---|
| alt.atheism | god | 16.9 |
| comp.graphics | graphics | 22.3 |
| comp.os.ms-windows.misc | windows | 46.0 |
| comp.sys.ibm.pc.hardware | drive | 24.2 |
| comp.sys.mac.hardware | mac | 25.1 |
| comp.windows.x | window | 27.8 |
| misc.forsale | sale | 24.4 |
| rec.autos | car | 37.7 |
| rec.motorcycles | bike | 31.0 |
| rec.sport.baseball | year | 18.1 |
| rec.sport.hockey | game | 24.9 |
| sci.crypt | key | 36.2 |
| sci.electronics | use | 10.6 |
| sci.med | msg | 14.7 |
| sci.space | space | 33.7 |
| soc.religion.christian | god | 45.9 |
| talk.politics.guns | gun | 25.8 |
| talk.politics.mideast | israel | 30.5 |
| talk.politics.misc | people | 14.5 |
| talk.religion.misc | god | 15.1 |

Table 5: tf-idf score over most important word in each 20Newsgroups dataset class

| Class | 10 Most important word | tf-idf scores |
|---|---|---|
| 0 - Negative | just | 76.6 |
| | work | 76.1 |
| | day | 58.1 |
| | like | 56.7 |
| | today | 54.6 |
| | don | 52.8 |
| | going | 51.1 |
| | sad | 49.6 |
| | really | 49.0 |
| | miss | 49.0 |
| 4 - Positive | good | 84.9 |
| | just | 77.0 |
| | love | 66.3 |
| | thanks | 62.5 |
| | day | 61.7 |
| | http | 56.1 |
| | lol | 54.4 |
| | going | 53.6 |
| | quot | 53.2 |
| | time | 50.1 |

Table 6: tf-idf score over most 10 important word in each Sentiment140 dataset class