COMP 551: Mini-Project 2

Antoine Bonnet
260928321
McGill University

Dragos Secrieru
260923248
McGill University

Cyril Saidane
260706157
McGill University

Abstract

The goal of this project is to apply different linear classification machine learning models and compare their respective performance on two document classification benchmark datasets. The Naive Bayes (NB) and Logistic Regression (LR) models were simulated on both datasets to obtain the optimal models with maximal validation accuracy through hyperparameter tuning. Through our experiments, we obtained three major findings. First, we found that the NB model outperforms the final test accuracy of LR on both datasets. Second, we obtained a significantly higher performance of both models on the binary classification dataset than on the dataset comprising 20 different classes (by a margin of more than 10%). Finally, we observed that NB runs significantly faster compared to LR (2 vs. 30 minutes) on both datasets.

1 Introduction

1.1 Motivation and findings

The main goal of this project was to compare two different classification models on two different datasets. Both datasets contain instances of text text with assigned class labels. The two main differences between the datasets were their size (NewsGroups is ten times smaller than Sentiment140) and their number of different classes (Newsgroups has 20, while Sentiment140 has 2).

This discrepancy was reflected in the final performance of the models, since the average test accuracies were approximately 10% lower for the NewsGroup dataset than for the Sentiment 140 dataset, due to the higher number of classes.

After performing hyperparameter tuning on both models, we achieved test accuracies of 69.3% and 67.8% with NB and LR respectively on the NewsGroups dataset and test accuracies of 79.7% and 77.7%, with the same models, on the Sentiment140 dataset.

1.2 Relevant work

Prior to our experiments, we found a paper (1) discussing several cleaning techniques for the Newsgroups dataset. They discussed how stemming and stop word removal are amongst the most important techniques, which inspired us to use them. Their main advantage seem to be in decreasing the file size, thus increasing runtime performance.

Using Naive Bayes, (2) achieved an F_1 score of 0.91 while (3) achieved an accuracy of 59% on the Sentiment140 dataset. We also found that (4) achieved an accuracy of 83% on the Sentiment140 dataset using logistic regression and pre-processing the dataset with tf-idf weighting. This suggests that tf-idf pre-processing was worth considering, given the strength of the results found. In all four papers, emphasis is put on the importance of data pre-processing. The non-numerical nature of these textual datasets makes data processing a necessary step to render the data digestible by the classification models.

2 Datasets

The 20 Newsgroups dataset contains 18K newsgroups posts each labelled by one of 20 different topics, split into training and test sets of size 11314 and 7532 respectively. Class counts are reasonably uniform; for the training set, the class counts range from 377 to 600.

On the other hand, the Sentiment140 dataset contains 1.6M Tweets, each categorized as having positive or negative sentiment, with a much smaller test set comprising 359 instances. As this training set was too large for our models, we truncated it by an order of magnitude, giving us a training set of 160K instances. Its class distribution is almost uniform (80052 negative instances). The Sentiment140 dataset therefore contains more instances, but fewer classes than the Newsgroups dataset.

2.1 Pre-processing features

The pre-processing steps were uniformly applied on both datasets to increase the accuracy of classification models. We followed the "bag of words" approach. First, each document string was converted to lowercase, stripped away of accents and common English stop words were removed (words with limited meaning for the purpose of classification). We then implemented stemming using the classic porter algorithm provided by the NTLK library. This algorithm cuts words down to their most basic form to ensure that related words correspond to the same number after processing. Words like "familiar" and "familiarize", for example, are cut down to "famil". This reduces the size of the vocabulary and makes classification easier by removing redundant or useless features.

Then, the datasets were scanned and all the words used in them were recorded into a corpus. Each instance in the data was turned into a lengthy matrix of numbers where

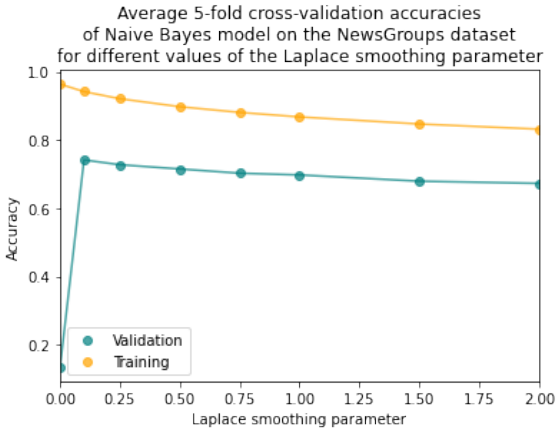

Figure 1: NB performance on Newsgroups dataset

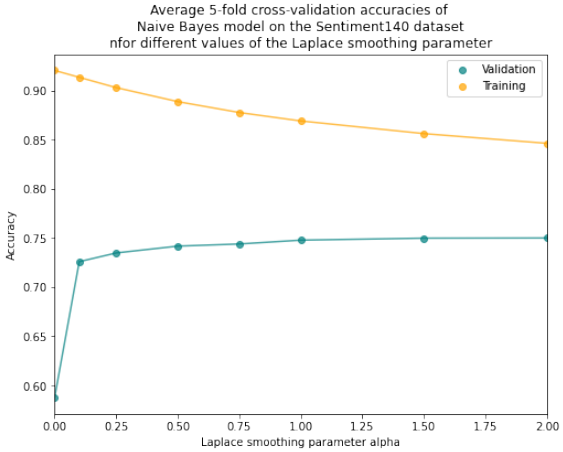

Figure 2: NB performance on Sentiment140 dataset

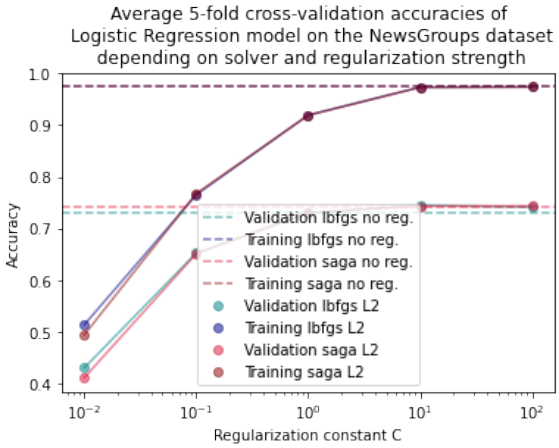

Figure 3: LR performance on Newsgroups dataset

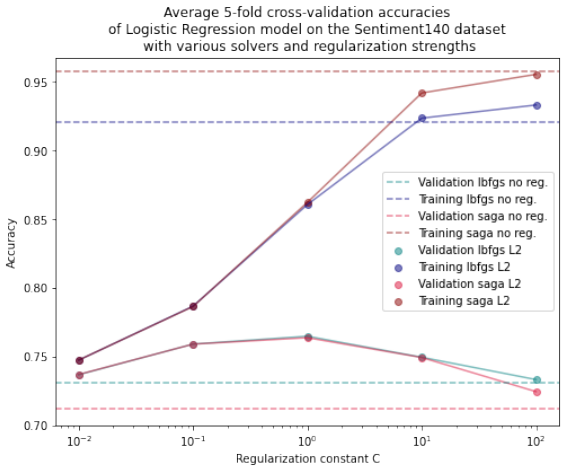

Figure 4: LR performance on Sentiment140 dataset

each number corresponds to the count of occurrences of a given word from the corpus in that instance. We ensured the test set followed the same vocabulary as the training set to maintain consistency.

Word counts were then scaled by term frequency-inverse document frequency (tf-idf) . This method gives less predictive weight to common words which appear in high frequency in the corpus. This was done because we expect a small number of highly specific words to be most correlated with class labels. These words show in lower frequency, while high frequency words carry less important information for the purpose of classification.

3 Results

In order to pick the best model, hyperparameter tuning was performed for each model on each dataset. For every set of hyperparameter values, the average 5-fold cross-validation training and validation accuracies of both models were computed (see Sections 3.1 and 3.2). We then chose the optimal hyperparameter as those yielding highest average validation accuracy over all 5 splits. We then trained our model with optimal hyperparameters on the whole training set, then applied it on the reserved test set to obtain the final test accuracy (see Section 3.3).

3.1 Multinomial Naive Bayes

In accordance to the nature of the data provided, we implemented a Multinomial Naive Bayes model with Laplace smoothing hyperparameter $\alpha \geq 0$ (see Section 6 for details). For each hyperparameter value, the corresponding average 5-fold cross-validation accuracies of the NB model on the Newsgroups and Sentiment140 datasets are respectively shown in Figures 1 and 2.

For both datasets, $\alpha = 0$ yields both the best training accuracy and worst validation accuracy. For the Newsgroups dataset, the $\alpha = 0.1$ parameter yields the highest validation accuracy of 74.1% (see Figure 5).

Dataset	NewsGroups		Sentiment140	
Model	Naïve Bayes	Logistic Regression	Naïve Bayes	Logistic Regression
Optimal parameter	$\alpha = 0.1$	solver = lbfgs, penalty = l2, C = 10	$\alpha = 2$	solver = lbfgs, penalty = l2, C = 1
Maximal validation accuracy (%)	74.1	74.5	75.0	76.5
Training accuracy (%)	94.2	97.3	84.6	86.1
Test accuracy (%) (with pre-processing)	69.3	67.8	79.7	77.7
Test accuracy (%) (without pre-processing)	65.9	63.9	80.5	81.0

Figure 5: Optimal hyperparameters and corresponding average 5-fold cross-validation accuracies for both models on each dataset (with and without tf-idf/stemming pre-processing)

On the other hand, the NB validation accuracy on the Sentiment140 dataset increases with the value of α, and so the optimal parameter value was therefore $\alpha = 2.0$ with maximal validation accuracy 75.0%. In future experiments, higher values of α might be tested to obtain higher validation accuracies, but they seemed to follow an asymptotic behaviour.

In absolute terms, the training accuracies reached extremely high levels (up to 95% with low α), while the validation accuracies remained generally constant at reasonable levels (near 75% for both datasets for $\alpha > 0$). Multinomial Naive Bayes seems to hold the similar levels of performance over both datasets, and so it appears to be a robust linear classification model.

3.2 Logistic Regression

We also applied the Logistic Regression model (provided by Sci-kit learn) with hyperparameters being the 'solver' (i.e. the optimization algorithm used by the model), as well as the regularization parameter C ranging from .01 to 100 (which corresponds to the inverse of the strength of ℓ_2 regularization $\lambda(C) = 1/C$). We used the 'lbfgs' solver (see Limited-Memory Broyden–Fletcher–Goldfarb–Shanno algorithm) and the 'saga' solver (an extension of the Stochastic Gradient Descent algorithm).

We simulated both solvers with regularization for various values of the regularization parameter C, as well as without regularization. The corresponding average 5-fold cross-validation accuracies of the LR model on the Newsgroups and Sentiment140 datasets are respectively shown in Figures 3 and 4.

For the Newsgroups dataset, both training and validation accuracies increase with higher values of the C parameter, until it reaches a plateau which meets the accuracy values for the models without regularization. This makes sense because higher values of C yields lower values of the regularization strength, as $\lim_{C\to\infty} \lambda(C) = 0$ until it converges to no regularization. This increasing trend suggests that weaker regularization yields higher accuracy for the Newsgroups dataset. As for the optimization algorithms, both solvers 'lbfgs' and 'saga' yielded almost identical accuracies, while 'saga' had a 20% faster running time than 'lbfgs' on average. For the Newsgroups dataset, the optimal set of LR hyperparameters was the 'lbfgs' solver with ℓ_2 regularization constant $C = 10$ with an average validation accuracy of 74.5% (as shown in Figure 5).

For the Sentiment140 dataset, the training accuracies were strictly increasing with regards to C, but in this case the validation accuracies formed a parabolic shape with global maximum located at $C = 1$. The corresponding optimal hyperparameters were therefore given by the 'lbfgs' solver with ℓ_2 regularization constant $C = 1$ with average validation accuracy of 76.5%.

Logistic Regression solvers failed to converge in most cases on both datasets for values $C > 1$, i.e. with low regularization strength $\lambda < 1$. This led to significantly higher running times. However, this divergent behavior did not necessarily lead to low accuracy, as we can see in the case of Newsgroups that higher values of C led to higher validation accuracy.

3.3 Model Performance

Using the optimal hyperparameter values found in sections 3.1 and 3.2 (Laplace smoothing parameter α for Naive Bayes; solver and regularization penalty C for Logistic Regression), we trained both NB and LR models on the whole training set and used it on the test set to obtain final test accuracies as shown in Figure 5.

We found that NB consistently displays an improvement of approximately 2% over LR regardless of the dataset.

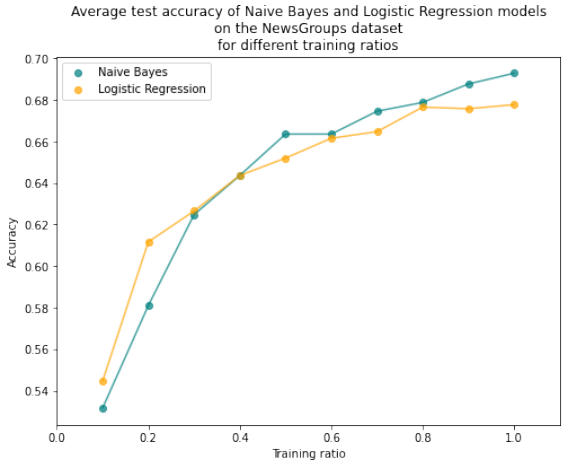

Figure 6: Model accuracy on Newsgroups dataset

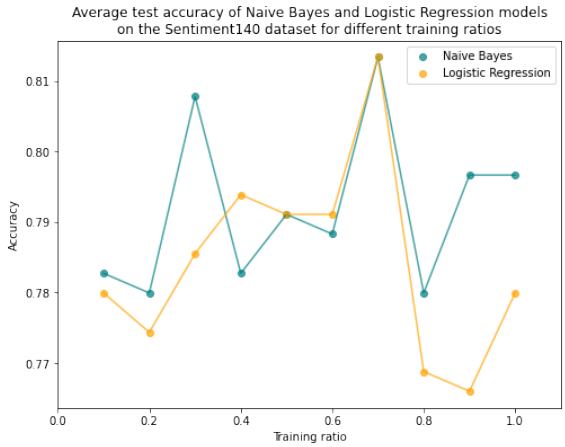

Figure 7: Model accuracy on Sentiment140 dataset

Comparing the final test accuracies of each model when varying the training ratio (the proportion of training data used) from 10% to 100% produced interesting results. For the Newsgroups dataset, larger training ratio yielded higher model performance, as expected (see Figure 6). On the other hand, for the Sentiment 140 dataset, we see that no matter the training ratio the accuracy is quite unstable and stays to similar levels between 76% and 81% for both models. This is caused by the large size of the Sentiment140 dataset, which means that for any training ratio higher than 10%, the training set used is large enough to produce good results.

3.4 Feature selection

We wanted to explore the effects of performing feature selection through tf-idf weighting and stemming during the pre-processing stage. To do so, we ran the exact simulations as before, but without these processing steps, and reported the final test accuracies in Figure 5.

We found that pre-processing the Newsgroups dataset increased the final test accuracy by about 4% on both models. On the other hand, for the Sentiment140 dataset, pre-processing decreased the final test accuracy by 0.8% for NB and 3.3% for LR. We cannot therefore reach a conclusion on the effects of tf-idf weighting and stemming, as they do not yield consistent effects across datasets.

3.5 Naive Bayes predictive strength

We analyzed the predictive strength of the Multinomial Naive Bayes model on the Newsgroups dataset by training the model with optimal hyperparameter $\alpha = 0.1$ on the entire training set. We then computed the rank of the predicted posterior probability of each true label among all 20 classes of the Newsgroups dataset. The rank distribution is shown in Figure 8. We fitted an exponential distribution with MLE $\hat{\lambda} = 1.275$, with variance $1/\hat{\lambda}^2 = 0.615$. In particular, we observed that the true class label is among the top 3 classes with highest posterior probability in 85% of instances, and among the top 5 in 90.5% of cases.

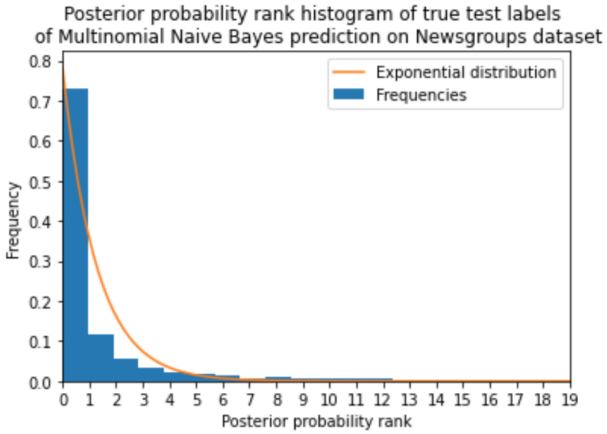

Figure 8: Multinomial Naive Bayes predictive strength

4 Discussion and Conclusion

To conclude, we found that Multinomial Naive Bayes is both faster and yields more accurate document classification than logistic regression for both datasets. We also found that the Sentiment140 dataset yields accuracies that are 10% higher than those of the Newsgroups dataset for both models.

We found 3 possible changes worth exploring for future experiments. First, we could explore other smoothing methods than Laplace smoothing. Secondly, we could only consider two classes in Newsgroups to see whether our test accuracy approaches that of Sentiment140, or whether the two datasets are fundamentally different. This would mean that the 10% accuracy gap stems from some other factor, perhaps the structure of the text itself. Lastly, additional

pre-processing techniques could be applied to the datasets. Since it's a text, there are many more techniques we could have attempted such as removing nametags or varying the n-gram range of the count vectorizer method.

5 Statement of Contributions

The workload was evenly distributed among the 3 team members. Here are their respective contributions:

Dragos Secrieru: Loaded the datasets, took part in pre-processing the data, ran simulations, designed experiments, wrote the 1,2 sections of the report.

Antoine Bonnet: Implemented Naive Bayes model, the K-fold cross validation method and the hyperparameter tuning method, ran simulations, created accuracy plots and wrote the 3,4 sections of the report.

Cyril Saidane: Set up the notebook collaboration tools, loaded datasets, cleaned data, ran pre-processing procedures (tf-idf, stemming) and wrote the 5,6 sections of the report.

6 Appendix

6.1 Multinomial Naive Bayes

Let $x_d^{(n)}$ denote the number of times word d appears in the n-th string and $y^{(n)}$ denote the label associated to the n-th string. The Naive Bayes model learns the class prior $p(y)$ and class conditionals $p(x_d|y)$ from the data to obtain the posterior class probability through Bayes' rule

$$\tilde{p}(y|x) = p(y)p(x|y) \propto p(y|x) = \frac{p(y)p(x|y)}{p(x)}.$$

The class prior represents the probability of each of the C classes, which corresponds to the class label frequencies in the dataset. This is given by the Categorical distribution:

$$p(y = c; \pi) = \prod_{c'=1}^{C} \pi_{c'}^{\mathbb{I}(y=c')} = \pi_c.$$

The class conditionals represent the probability distribution of each feature x_d given the class y. As our features represent word counts, we use the Multinomial distribution (hence the Multinomial Naive Bayes denomination):

$$p(x_d|y = c; \theta) = \frac{(\sum_{d=1}^{D} x_d)!}{\prod_{d=1}^{D} x_d!} \prod_{d=1}^{D} \theta_{d,y}^{x_d}.$$

The Naive Bayes model learns from the data from the training set by computing the maximum likelihood estimator (MLE) for both parameters $\hat{\pi}$ and $\hat{\theta}$. In practice, we add a Laplace smoothing parameter α to these computations in order to avoid zero probability events. The MLE for π corresponds to the frequency of class c among the dataset and so is computed as

$$\hat{\pi}_c = \frac{N(y = c) + \alpha}{N + \alpha C}.$$

On the other hand, the MLE for θ is computed as

$$\hat{\theta}_{d,c} = \frac{\alpha + \sum_n x_d^{(n)} \mathbb{I}(y^{(n)} = c)}{\alpha D + \sum_n \sum_{d'} x_{d'}^{(n)} \mathbb{I}(y^{(n)} = c)}$$

Once the Naive Bayes model is trained, we can predict class labels $\hat{c}$ for new instances x by maximizing the log-posterior

$$\begin{aligned} \hat{c} &= \operatorname*{argmax}_c p(y = c|x) \\ &= \operatorname*{argmax}_c \log \tilde{p}(y = c|x) \\ &= \operatorname*{argmax}_c \log(\hat{\pi}_c) + \sum_{d=1}^{D} x_d \log \hat{\theta}_{d,c} \end{aligned}$$

To convert this unnormalized log-posterior $\log \tilde{p}(y = c|x)$ to a probability $p(y = c|x)$, we normalize the data using the log-sum-exp trick.

References

[1] Albishre, K., Albathan, M., amp; Li, Y. (2015). Effective 20 newsgroups dataset cleaning. 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT). https://doi.org/10.1109/wi-iat.2015.90

[2] Xu, S. (2016). Bayesian naïve Bayes classifiers to text classification. Journal of Information Science, 44(1), 48–59. https://doi.org/10.1177/0165551516677946

[3] Goel, A., Gautam, J., amp; Kumar, S. (2016). Real time sentiment analysis of tweets using naive bayes. 2016 2nd International Conference on Next Generation Computing Technologies (NGCT). https://doi.org/10.1109/ngct.2016.7877424

[4] W. Habib, M., N. Sultani, Z. (2021). Twitter Sentiment Analysis Using Different Machine Learning and Feature Extraction Techniques. Al-Nahrain Journal of Science, 24(3), 50-54. Retrieved from https://anjs.edu.iq/index.php/anjs/article/view/2372