Project 2 - Group 62

Mamoun Benchekroun, Ines Moreno, Julien Mounthanyvong

March 18, 2020

1 Abstract

In this mini-project we explored text classification methods and compared the performance of seven machine learning models. We used two distinct data sets to evaluate these models: 20 news group data set and IMDB data set. The accuracy and the logarithmic loss entropy were the two metrics we used for assessing the performance of our models. Additionally, we compared the effect of tuning different parameters on each model in order to determine the ones that yield to the best performance. Overall, we found that linear models such as LR and NB obtained a higher average accuracy in both training and validation sets for both data sets. Tree approaches seemed to be the least strong.

2 Introduction

In this project, we were given two text data sets and we tested the performance of seven machine learning models through various experiments. The models tested were Logistic Regression (LR), Decision Trees (DT), Support Vector Machines (SVM), Adaboost (AB), Random Forests (RF), Complement Naive Bayes (CNB) and Multinomial Naive Bayes (MNB). The two data sets respectively deal with newsgroups posts on different topics and, movies reviews. The first data set was the support for multi-class classification and the second was a binary task. For the pre-processing of the data, we tokenized it and removed all stop-words. Then, we used the scikit-learn library to obtain the different algorithms for the models to test and finally, we created a set of functions to tune some parameters on the models and pick the ones optimizing its performance. We found that linear models such as NB, LR and SVM generally perform better than tree-based approaches like Adaboost, DT or RF.

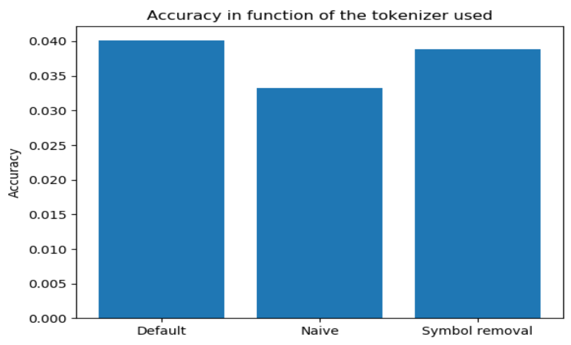
3 Related work

Many previous authors have explored the multi-class classification problem and text classification tasks using the same data sets than us. Aggarwal et al [1] gave a review of the different text classification algorithms and showed what are the advantages and disadvantages of each approach. According to Joaquims [6] text data is ideally suited for SVM classification because of its sparse high-dimensionality. Indeed, Drucker et al [4] showed that SVMs have the best performance when it comes to email spam classification compared to decision trees for example. On the other hand, decision trees have been shown to work best when in conjunction with boosting techniques. [5]. Moreover, Aly gave a review of the different multi-class classification methods [2].

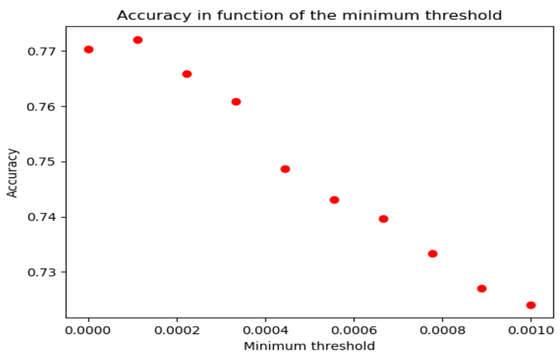
4 Data sets and setup

4.1 20 newsgroup data set

This data set is a collection of roughly 20,000 newsgroup documents partitioned across 20 different newsgroups. The samples are pretty evenly distributed across each class (see graph in annex). To find a good preprocessing method, we used Complement Naive Bayes to perform tests as it had pretty good results with the default parameters and it is a pretty fast classifier. We found that using unigrams led to better results than bigrams. Using both unigrams and bigrams also slightly improved the results, but we considered that this improvement wasn't significant enough to justify increasing the feature space that much. We also found that the default tokenizer was more fitted than the custom one we proposed, and that using scikit-learn stop-words list led to better results. Finally, we saw that it was better to use a very small minimum threshold (concerning the frequency of a term appearing in a document), while using a maximum threshold did not have any impact.



(a)



(b)

Figure 1

4.2 IMDB data set

This data set, provides us with 50,000 movies reviews labeled as positive or negative. The goal is to predict the sentiment of a review, so whether the review is positive or negative. Both the training and the testing set are composed of half positive reviews, and half negative ones. We began by using pre-processing steps similar to the ones used for the first dataset. We globally found that each parameters had about the same impact as before. For example, the tokenizer didn't have much impact, while decreasing the minimum threshold led to better results. In addition to these tests, we also used the Porter stemmer on the original texts to get more relevant features.

Due to a lack of space for this report, we will be mainly focusing on the first data set (the 20 newsgroup one). As such, we won't be including some graphs concerning the both data sets. Please refer to the graph included in the .zip document for that purpose.

5 Proposed approach

We tested a total of seven different models, tuning their hyper-parameters. The models we studied were the following:

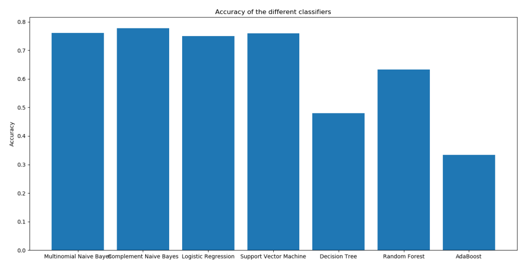
- **Logistic Regression (LR)**: a model that uses the logistic sigmoid function to return a probability value which can be mapped to two or more classes.
- **Decision Trees (DT)**: a model where a decision tree is used to make predictions on how to go from an observation (represented in the branches) to a class label (represented in the leaves) in order to classify elements.
- **Adaboost (AB)**: a model that trains a weak learner on a dataset repetitively, giving more weight to the samples were the learner was wrong at each iteration.
- **Support Vector Machines (SVM)**: a supervised learning model that uses a representation of data as points in space mapped so that the data points belonging to different classes are divided by a clear gap. New data points are then mapped into the same space and predicted to belong to a specific class depending on which side of the gap they fall into.
- **Random Forests (RF)**: a model that uses different decision trees, each fitted on a subset of the dataset, and average their result in order to give a final prediction
- **Naive Bayes (CNB and MNB)**: The MNB model is usually used for classification with discrete features (word counts) as well as the CNB model which is very similar to MNB but was designed to correct the 'severe assumptions' made in MNB.

To evaluate the performance of each model, we used 2 different indicators. The main measure was the accuracy, and we aimed to optimize this value in our work. We also included the log cross entropy since even though it does not measure the actual performance of our model, it indicates the level of purity of the groups created by our classifiers. The data sets were stored as sparse matrix as most sample would only contain a few number of different words compared to the overall vocabulary, and such a format also allowed us to shuffle the data set for a better training.

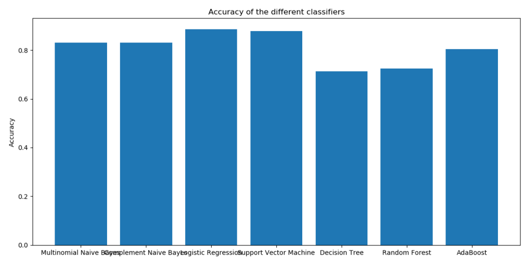
6 Results

Here we report the results we obtained for each of the models tested. Note that for the hyper parameters selection, we decided not to use grid-search as it was too computationally heavy. Thus, we only used grid search to fine tune the best performing model. Instead, we studied the effect of each parameter independently. Moreover, we chose to only study subset of parameters that could be used not only for binary classification but also for multi-class classification.

6.1 Model Comparison



(a) 20 newsgroup training set



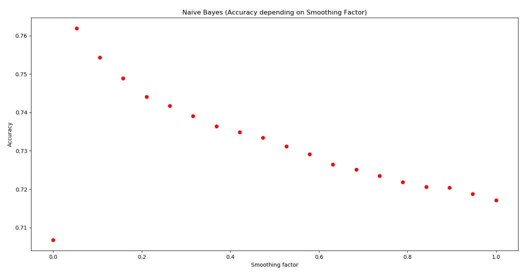
(b) IMDB test set

Figure 2

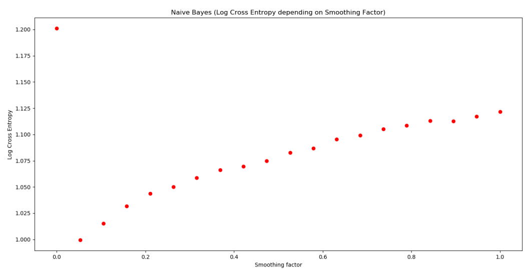
We can see that tree-based models (DT, RF, AB) generally perform worse than the others. This might be because we didn't use any boosting for DT or RF. Furthermore, as studied by Aggarwal, SVM and Naive Bayes methods are generally more robust in terms of performance. [1]

6.2 Multinomial and Complement Naive Bayes

In accordance with Colas et al. [3], NB approaches seem are the best performing when they have a large number of features: with CNB at the top of the podium and MNB following it for the 20 newsgroup data set. For both models, we decided to test different smoothing factors (alpha) to see how it will impact our accuracy. As we can see in Figure 1, we found that the accuracy of MNB decreased with a larger smoothing factor. Parallel to this, the log cross entropy increased meaning an overall worse performance. Hence, we decided to keep a relatively small smoothing factor (alpha = 0.01). In the case of CNB nevertheless, increasing the smoothing factor didn't influence much the accuracy of the model. This was the case too for both models on the IMDB data set.



(a) 20 newsgroup



(b) 20 newsgroup

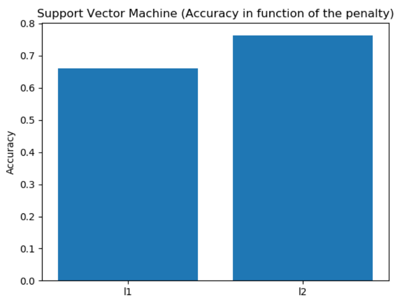
Figure 3

6.3 Logistic Regression

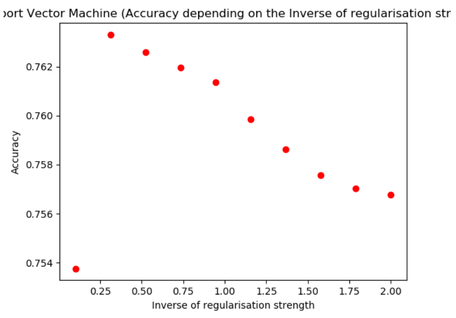
This model performed fairly well: 0.745 accuracy on the multi-class classification for the 20 newsgroup data set and 0.89 accuracy on the IMDB data set. We decided to test various hyper-parameters: epsilon, number of iterations and regularisation but none of this hyper-parameter affected considerably the performance of the model, though the accuracy did change depending on the regularisation strength.

6.4 Support Vector Machine

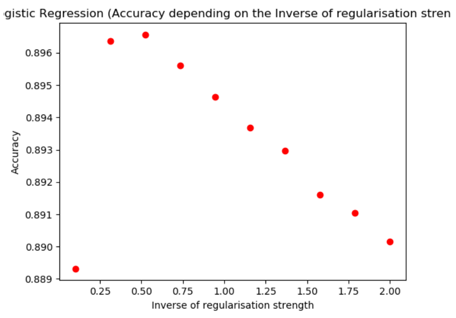
Inspired by Colas et al [3], for this model we tested different epsilon values, number of iterations, penalties, loss function and regularisation strengths. While the epsilon values and the number of iterations didn't seem to affect our model's accuracy, we found that the best penalty was the default L2 (Figure 2a) and that, the model performed worse when we increased the regularisation strength (or decreased its inverse Figure 2b).



(a) 20 newsgroup



(b) 20newsgroup

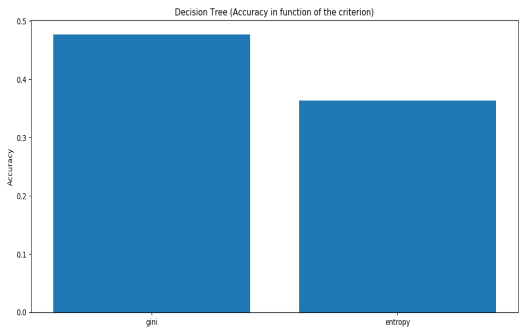


(c) IMDB

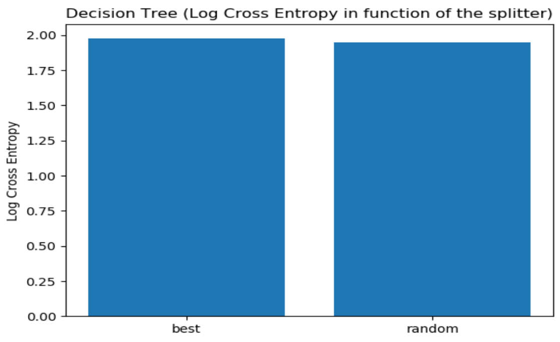
Figure 4

6.5 Decision Tree

For this model, we tested the criterion and the splitter hyper-parameters. We observed that the best criterion was the default gini impurity (Gini impurity measures how often a randomly chosen element is misclassified) and, as for the splitter (strategy used to choose the split at each node) we didn't observe much difference between the default best and the random. This must be because the default 'best' split is also random.



(a)

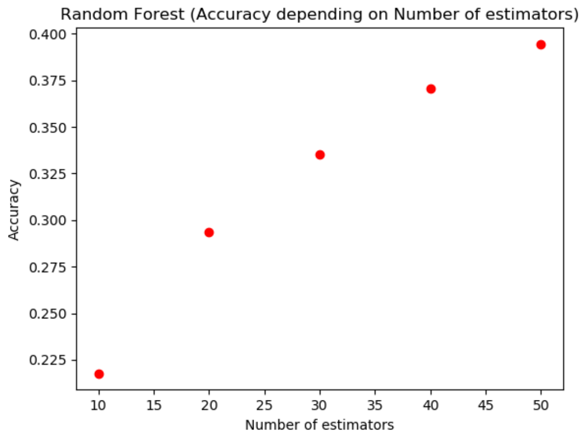


(b)

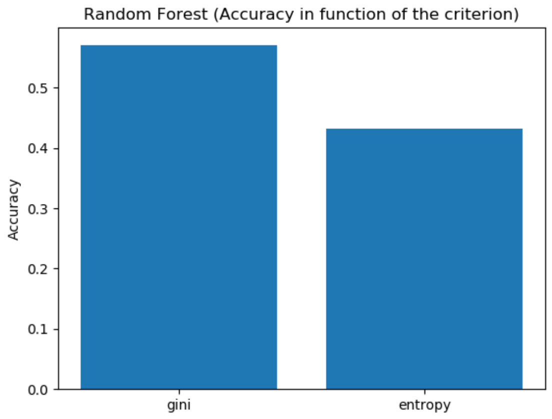
Figure 5

6.6 Random Forest

Here, we tested the performance of the model with different number of estimators and with different criterions. We found that the accuracy increased with the number of estimators (Figure 2a) and that the best criterion was the default gini impurity (Figure 2b). Nevertheless, the log cross entropy was slightly better using the best split. Moreover, increasing the number of estimators significantly slowed down the model, so there was a trade-off between the speed and the performance.



(a)

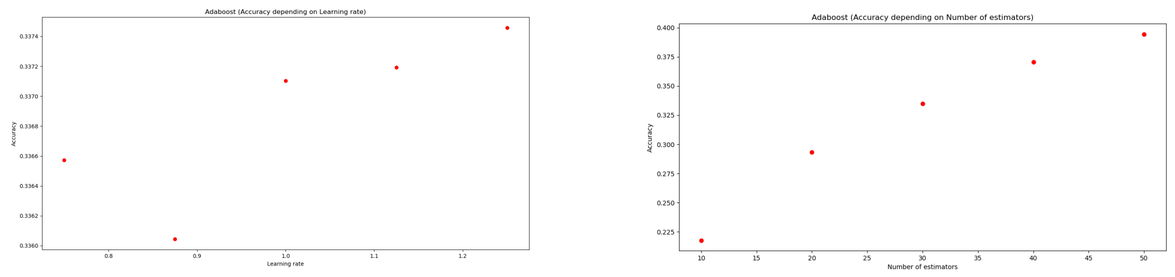


(b)

Figure 6

6.7 Adaboost

Here, we used a decision tree with depth 1 as our weak learner. We observed the performance of this model with different number of estimators and different learning rates. We found that the accuracy increased when we increased the number of estimators (Figure 2b), and we found our best learning rate at 1.25 (Figure 2a). Ideally, we know that we could get better results by increasing the number of estimators and taking a low learning rate, but we did not do so as it would really slow down the model. The low number of estimators (30) might be a big reason of the low performance of our adaboost model.



(a) (b)

Figure 7

IMDB dataset

We ran similar tests on the IMDB dataset and overall, we got pretty similar results regarding the impact of each hyper-parameters. However, we do remark that overall, changing the hyper parameters tend to have a lesser impact than before.

Also, our models performed significantly better on the IMDB dataset, mostly due to the fact that a binary classification is easier than a multi-class one. On this dataset, the best performing model is logistic regression, followed by SVM.

7 Conclusion

Throughout this project we investigate and compare the performance of seven different models on two text classification tasks. The first task, with the 20 newsgroup data set was a multi-class classification and the second one, with the IMDB data set, a binary one. We compared the accuracy of all models on this two data sets using different methods and investigated which were the optimal parameters to use in each case to obtain the best performance. For their best performing hyper-parameter logistic regression and SVM remain at the top of the list for both datasets out of the five classifiers initally required. They are however matched and surpassed in performance by the Naive Bayes classifiers for the multi-class task.

	MNB	CNB	LR	SVM	DT	RF	ADA
20 Newsgroup	0.70	**0.71**	0.69	0.70	0.44	0.60	0.31
IMDB	0.83	0.83	**0.89**	0.88	0.71	0.73	0.80

For future work, we could hope to run tests that would take longer to find better optimisation of each model. For example, using grid search with each model would be a nice improvement. Notably, being able to study a wider array of parameters for tree models and for adaboost could prove to yield far better results.

Furthermore, we could still investigate additional parameters. We could for example, use a different kind of metric for performance as F1-Score for the binary classification. Or even try different models such as k-NNs.

8 Statement of contribution

We all contributed in all sections by reviewing / testing each other's code.

- Mamoun Benchekroun: Preprocessing + data analysis
- Ines Moreno: write up + tests
- Julien Mounthanyvong: classifiers + parameter tuning

8.1 Extra features

- Implementation of both Multinomial Naive Bayes and Complement Naive Bayes
- Analysis of the log cross entropy in addition to the accuracy

- Testing of different preprocessing parameters such as tokenizers, stop words list, ngram used, and thresholds.
- Analysis of the distribution in the 20 newsgroup dataset, use of a stemmer in the IMDB dataset

References

[1] Zhai C. Aggarwal C.C. “A Survey of Text Classification Algorithms”. In: *Mining Text Data*. Springer, Boston, MA, 2012.

[2] M. Aly. “Survey on multiclass classification methods”. In: *Survey on Multiclass Classification Methods* (Jan. 2005).

[3] Fabrice Colas and Pavel Brazdil. “Comparison of SVM and Some Older Classification Algorithms in Text Classification Tasks”. In: vol. 217. Aug. 2006. DOI: 10.1007/978-0-387-34747-9_18.

[4] Harris Drucker, Donghui Wu, and V.N. Vapnik. “Support Vector Machines for Spam Categorization”. In: *Neural Networks, IEEE Transactions on* 10 (Oct. 1999), pp. 1048–1054.

[5] Yoav Freund and R.E. Schapire. “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting”. In: *Journal of computer and system sciences* 55 (Aug. 1997), pp. 119–.

[6] Joaquims T. “Text categorization with Support Vector Machines: Learning with many relevant features”. In: *Machine Learning: ECML-98, Lecture Notes in Computer Science*. Vol. 1398. Springer, Berlin, Heidelberg, 1998.