ECSE 551 - Mini-Project 2

Anonymous Author(s)
Affiliation
Address
email

Abstract

The aim of the second mini project was to implement a Bernoulli Naïve Bayes multidimensional (BNB) classifier from scratch. The classifier's aim is to predict the source location of a subReddit post from a pool of 4 subreddits possible : Toronto, Montreal, Paris and London. Preprocessing was done on the text to get rid of useless features. In addition to our BNB classifier, we have used other off the shelf models from the sklearn Python library in order to achieve the maximum possible prediction accuracy. Finally, we have found that the model providing the most accurate prediction was the Stacking with Bagging classifiers, with an accuracy of 69.4%.

Contents

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

1 Introduction

The main goal of our project was to implement a Bernoulli Naive Bayes (BNB) classifier from scratch in order to classify posts on Reddit. The posts could be from the following subreddits: Toronto, Paris, Montreal and New York City. The dataset is made of a "train" CSV file with bodies and labels, and a "test" CSV file only containing bodies. After implementing the mandatory classifier, other models were implemented to see if we could get better accuracies, namely: Decistion Tree (DT), Logistic Regression (LR), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analaysis (QDA), Support Vector Machines (SVM) as well as Stacking along with Bagging. Accuracies were evaluated using the 10 folds Cross Validation method.

2 Dataset

2.1 Data Analysis

The training set is composed of 719 sample reddit post and comment, each from one of 4 subreddits : Montreal, Paris, London and Toronto. The distribution of each of these 4 classes is largely even, with the exact distribution of samples depcited in the figure below.

Figure 1: Value count for each of the four class

Below is a table portraying the distributions (in percentages) of the Reddit post by subreddit:

Table 1: Probability distributions of each classes in training data

Montreal	Paris	London	Toronto
0.25035	0.25035	0.25035	0.24896

2.2 Pre-processing

In order to be able to train models on the textual data input, some preprocessing had to be done. We would like to note that all of the following steps have been designed to be activated or not on an iterative approach, so that we could tailor our preprocessing of the data to the model implemented. The first step of this process was to get rid of all punctuation from all samples, as they do not provide any valuable information on the subreddit source. Then, the next step for us was to get rid of generic stop-words, which also do not provide much information as they are expected for all

subreddits, and therefore are not a differentiating factor. Then, we implemented 2 different tokenizers : a LemmaTokenizer based on the lemmatization of the data, and a StemTokenizer based on the stemming of the data. The aim of these tokenizers is to reduce the number of unnecessary features in our dataset, by only selecting for each analyzed word the root of that word (Lemmatization), or by truncating parts of that word (Stemming). We have also implemented 2 different types of vectorizers to vectorize the data : a TF-IDF vectorizer and a CountVectorizer. These vectorizers allow us to weight each analyzed word according to the importance of that word in the dataset, and in the specific sample in which it is analyzed. Finally, we have implemented the option of normalizing the data, which basically normalizes the value of each word in a sample with respect to the unit norm of the sample.

3 Proposed Approach

3.1 Bernoulli Naive Bayes

The first model implemented was the BNB. The Naive Bayes class was made based on lectures' content, namely Lectures 9 and 10. Laplace Smoothing was also incorporated to our class as an option, but we've noticed that accuracy was fairly higher when it was activated. 10 folds Cross Validation was run on our model along with text processing hyper-parameter tuning in order to improve accuracy. Iteration was done over activation of TF-IDF, activation of normalization, activation of Laplace Smoothing and number of features allowed to the text processor. Our best accuracy was 63.42 % and was surprisingly found when TF-IDF and Normalization were both deactivated, and 2500 features were fed to the text processor without any tokenizer. Given that Naive Bayes mostly relies on counting, there is no hyperparameter tuning to do and its accuracy cannot be improved to a greater extent. In the appendix can be found a depiction of the runtime of our iterations on our own Bernoulli Naive Bayes classifier depending on the number of features fed.This graph shows us the clear correlation between computational cost and number of features.

3.2 Decision Tree Classifier

In an effort to improve the efficiency of our model, we have implemented a decision tree. We have selected the off the shelf Decision Tree model from sklearn, and we have iterated over multiple hyper-parameters to tune it to our dataset. We have also iterated over which preprocessing was the most adapted to a Decision Tree Classifier, and found that it was one with punctuation removed, stop words activated, TF-IDF weighting as well as Lemma Tokenizer. The hyperparameters used to achieve the highest accuracy were a maximum depth of 12 and a number of features of 3000. Runtime execution graph is included in the Appendix.

3.3 Logistic Regression

We have implemented the logistic regression model from sklearn. We have also found iteratively that the most accurate method to preprocess the data for this model was to enable TF-IDF weighting, normalize the data set, set the maximum number of features to 3000, and not using any specific tokenizer.

3.4 LDA

Similarly to what was done with Logistic Regression, we have implemented the LDA model from the sklearn library, and found iteratively that the best method to preprocess the data was to enable TF-IDF weighting, normalization and setting the maximum number of features to 2000, while using a LemmaTokenizer.

3.5 KNN

We have implemented the KNN model from the sklearn library, and found that the bets way to process the data with that model was to enable TF-IDF weighting, normalization and setting the maximum number of features to 2500, while using a Stemming tokenizer. We have also found that the most accurate model on the training data was the 10 neighbours KNN.

3.6 SVM

Support Vector Machines were also implemented from the sklearn library. It was first used alone along with optimization of text processing parameters, and accuracy was evaluated using our own Cross Validation Class. The best results were found activating both TF-IDF weighting and normalization, feeding the classifier 2500 features using the Stemming tokenizer. Afterwards, SVM was implemented along with Bagging. Iterations were done over the number of features, the type of tokenizer and the kernel our SVM model was used.

3.7 Stacking

Our most accurate and complex model was the Stacking classifier, imported from the sklearn library. The Stacking classifier used the following estimators : Decision Tree, Random Forest, KNN, Logistic Regression, Gaussian and Multinomial Naive Bayes, and SVM. The final estimator was a Logistic Regression Model. All the estimators were Bagging Classifiers, meaning all estimators mentioned were all duplicated (5 times) and were trained independently using bootstrap sampling.

3.8 Bagging

In our project, Bagging Classifier was implemented from the sklearn library at two steps, namely when we were implementing SVM and Stacking classifiers. This allowed us to train independently the same classifiers on different instances, yielding out the best result through majority vote.

4 Results

A summary of our results can be found in the table below.

Table 2: Classifier Model Performance

Classifier	Model Accuracy (%)	Runtime (s)
BNB	63.4	27.8
DT	56.74	2.73
LR	61.97	2.92
LDA	57.75	4.64
KNN	63.38	17.1
Stacking	69.4	493.02

As depicted in the table above, The highest results were obtained for the stacking model, which was based on the following models stacked :Decision Tree, Random Forest, KNN, Logistic Regression, Gaussian and Multinomial Naive Bayes, and SVM.
Below is the confusion matrix obtained on our best model (Stacking):

We can observe that our classifier struggled to correctly classify samples from Montreal. Many explanations could be given: use of both French and English, Quebec slang ...
After going through the reviews, we noticed that many of the posts in the Montreal subreddit had nothing to do with Montreal, or any crucial

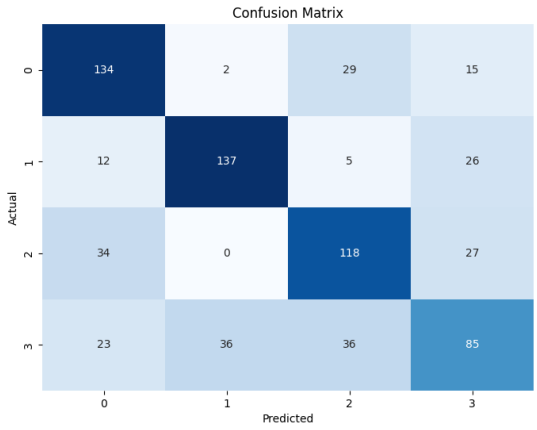

Figure 2: Confusion Matrix

5 Discussion and conclusion

The main aim of this assignment (MP2) was to develop a non-binary classifier to accurately predict the source location of some reddit posts, out of 4 possible locations : Montreal, Toronto, Paris and London.
In order to do so, we have started by building a Naive Bayes classifier from scratch. This classifier was then fed textual data whose vectorization, tokenization and normalization was fine tuned to optimize the accuracy of our model. This assignment highlighted the importance of the preprocessing of the data for each of the implemented models, as the selection of the correct preprocessing methods had a major effect on both the accuracy and the run time of our models. It also tells us how sensitive a model can be depending on the preprocessing done.
Furthermore, a key takeaway of MP2 was to understand how stacking and bagging can constitute powerful tools for the generation of robust classifiers. As a matter of fact, our classifier implementing stacking and bagging achieved the highest accuracy overall, and was therefore the one selected for the Kaggle competition.

Even though this model achieved the highest accuracy overall, it was also by far the most computationally expensive. Indeed, there was a clear trade-off for this model between the training time and the accuracy. One possible point of improvement of this model would be to implement more complimentary classifiers as opposed to rather similar classifiers, as these do not improve the accuracy of the prediction, in addition to making the stacking classifier more rigid and computationally inefficient. we were not able to iterate over them due to computation constraints. Furthermore, although we have implemented n-grams preprocessing, the implementation was not worth the slight gain in accuracy when there was any. One could also study the effect of a classifier that would be sensitive to feedback from the data to "record" common sequences of words, such as a Recurrent Neural Network for example.

6 Statement of Contributions

Hamza handled most of the pre-processing as well as implementations of selected models. Aymen implemented the Naive Bayes classifier from scratch as well as the Cross Validation model. Rest of the work was evenly shared.

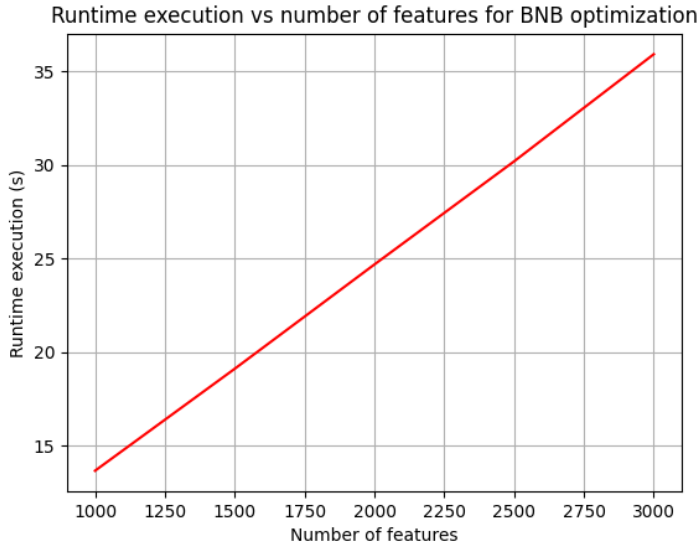

Figure 3: Runtime execution vs number of features for BNB optimization

Figure 4: Runtime execution vs max depth for DT optimization

Appendix

Note : In some parts of this report, classes (London, Montreal, Paris and Toronto) are also refered to as indices (0,1,2,3). Here is the correct indexation for this project : London = 0, Paris = 1, Toronto = 2, Montreal = 3.