Section X – Naïve Bayes

Section X.1 – Introduction

Naïve Bayes are a family of probabilistic classifiers based on Bayes' Theorem. These classifiers are simple and are applicable to many uses, such as document classification and spam filtering. Compared with other methods studied in this report, they are fast to train and, in theory, does not require much data to train on. Compared to neural networks, Naïve Bayes is a calculation, and thus less demanding in processing power and resource requirements.

For the project, three implementations of the Naïve Bayes was used. These are of the scikit-learn package as is the SVM implementation used. The first is the Gaussian Naïve Bayes, an implementation where the likelihood of the features is assumed to be Gaussian. Second is the Multinomial Naïve Bayes, where the data us assumed to be multinomial distributed. The multinomial implementation is one of the classic variants used for text classifications. The third and final implementation is the Bernoulli Naïve Bayes, suited for data with multivariate Bernoulli distributions, and would work well with binary valued data.[1]

The given test data was split up at the 11000 from last item mark, roughly two thirds of the given data. Everything up to the last 11000 entries are used as training data, given to the classifiers along with the label, and the remaining 11000 is used for validation testing, using the labels to check for errors. The data will be vectorized before being fed into the classifiers.

All three implementations were fed the same data, features, and, if applicable, tested with same settings, such as priors and smoothing settings. Tests were done with base settings (Title or description only, with no further tweaking), without smoothing applied, using titles along with categories, using titles along with categories but without smoothing, without using fit priors, and using with class priors. Class priors were given as [0.685,0.315] and [0.943,0.053] for clarity and concise respectively.

In addition, another feature list for testing conciseness, shared with the neural network tests, will be used as well.

Section X.2 – Gaussian Naïve Bayes

The Gaussian Naïve Bayes implementation was the first to be used. The implementation allows feeding in the prior probabilities of the class. Compared to the other Naïve Bayes implementations tests, Gaussian Naïve Bayes suffers from higher resource usage and is much slower. This is a consequence of requiring matrices be dense matrices. In order to get Gaussian Naïve Bayes to actually run, the virtual machine used to do the tests had to be given about 10gb of memory to work with, and even with this, applying Gaussian Naïve Bayes for the clarity tests failed due to not enough allocated memory. Because data batching was not done, results for Gaussian Naïve Bayes were skipped for the clarity tests.[2]

RMSE for GaussianNB	
Base	0.690
Title with categories	0.690
With class priors	0.690
Generated Features	0.577

The results of conciseness test were poor compared to the other implementations. The RMSE for the base fit was 0.690, and remained approximately the same across the tests using both titles and categories, and with class priors. Using the features shared with the neural network tests, the results improved to 0.577.

Section X.3 – Multinomial Naïve Bayes

The implementation for Multinomial Naïve Bayes performed much better than the Gaussian implementation. To start, it did not require matrices being fed in to be dense, and as such sparse matrices works. Unlike the Gaussian implementation, Multinomial Naïve Bayes actually has a smoothing function, and allows the toggling of fit priors.[3]

Because of the sparse matrices, training and predicting are also much faster. And even more, it was able to complete the clarity tests as well. For the concise tests, base results without medications resulted in the best RMSE results at 0.460. Multinomial Naïve Bayes performed very well with the clarity tests, clocking 0.275 as its best RMSE result, on base settings. Turning off smoothing and adding features does not seem to help in terms of improving results, nor did disabling fit priors and providing class priors. The generated features does not work with Multinomial Naïve Bayes due to having non-discreet negative values and as such was skipped.

RMSE for MultinomialNB	Concise Tests	Clarity Tests
Base	0.460	0.275
Without smoothing	0.491	0.362
Titles with categories	0.496	0.311
Titles with categories without smoothing	0.508	0.447
Without fit priors	0.478	0.320
Without class priors	0.501	0.401
Generated Features	No Data	

Section X.4 – Bernoulli Naïve Bayes

The last Naïve Bayes implementation tested was the Bernoulli Naïve Bayes. Like Gaussian Naïve Bayes, Bernoulli Naïve Bays ran into issues with exceeding available memory, although this time it was solved by using a hashing vectorizer instead of the count vectorizer as used by both Gaussian and Multinomial implementations. Training and prediction speed was faster than Gaussian, and not too much different from Multinomial. Like the Multinomial implementation, it has a smoothing function, and allows the toggling of fit priors in addition to being able to use class priors.[4]

Like the Multinomial Naïve Bayes implementation, Bernoulli Naïve Bays didn’t result in any significant RMSE results for the clarity tests, but did do well with the concise test. On average, the results for the clarity tests were higher than those provided by Multinomial, with the best result coming from turning off smoothing. For the clarity test, Bernoulli Naïve Tests achieved the best results out of the three implementations scoring 0.237 at the lowest using only the title and title with categories.

RMSE for BernoulliNB	Concise Tests	ClarityTests
Base	0.560	0.237
Without smoothing	0.479	0.384
Titles with categories	0.560	0.237
Titles with categories without smoothing	0.508	0.401
Without fit priors	0.560	0.751

Without class priors	0.560	0.751
Generated Features	0.528	

Section X.5 – Conclusions with Naïve Bayes

While Naïve Bayes, with the setups as done in this report, did not give good results for clarity test, they did decently well with the concise tests. These numbers are, however, still unmatched by results from our tests with SMV and TensorflowNN, both of which provided much better results in the sub 0.1s. TensorflowNN in particular, reached an MSE result of 0.045. However, compared to Tensorflow as will be shown, Naïve Bayes can give its results with significantly less resources required and time spent.

An interesting observation is that under almost all circumstances, using the base settings alone resulted in some of the best predictions available from Naïve Bayes, and that adding priors or other features such as categories into the training only serves to worsen the results. Parts of this could be argued that the requirements for scoring these tests didn't really look into the accuracy of the titles and descriptions themselves, and other attributes such as length may play bigger role in being a good determinant for predictions.

The rather high results from the tests could also indicate that Naïve Bayes may not be the most suitable method for the given task, lacking in concepts that a neural network may be able to learn and adapt to.

All the tests done with Naïve Bayes had training and prediction done within or around one minute, and within the 10GB memory cap (with the exception of Gaussian Naïve Bayes which exceeded 10gb for the concise tests). Compare this to Tensorflow's 20 minute training time and reaching 40gb in data requirements and the trade-offs are clear.

Section X.6 – Future work with Naïve Bayes

There are still more feature combinations that could be tested to see if results can be further improved. What was tested in this report is nowhere near exhaustive and other derivable features such as text length and combinations of titles and descriptions can be looked into. Different feature lists can be generated and used to as well to see the results of various combinations.

While it can be argued that the method itself is not suitable for the given tasks, it doesn't mean it's not possible to improve the results, as even if the prediction is not as good as other methods, its speed and resource requirements can still lend it an upper hand under conditions where speed is required and resources lacking.

References:

[1]Scikit-learn developers. 1.9. Naïve Bayes. Retrieved March 16, 2018 from http://scikit-learn.org/stable/modules/naive_bayes.html

[2]Scikit-learn developers. GaussianNB. Retrieved March 16, 2018 from http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[3]Scikit-learn developers. MultinomialNB. Retrieved March 16, 2018 from http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

[4]Scikit-learn developers. BernoulliNB. Retrieved March 16, 2018 from http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html