Literature Review - Bernoulli Naïve Bayes

Besides the multinomial Naïve Bayes algorithm introduced in class, in this project we take a close look at another kind of Naïve Bayes: Bernoulli Naïve Bayes, which differs from multinomial classifier in the representation of features. For Bernoulli classifier, binary numbers 1 and 0 are used to resemble the occurrence of one word, while multinomial one use integers to show the occurrence frequency. [1] explicitly states the calculation process of condition probability of one certain sample in prediction stage. Suppose a text sample is discretized into a binary vector in the data preprocessing stage with the words considered as features. The conditional probability of given that belongs to category can be expressed in the following formula:

From the formula, we can find that when the features are represented in binary, Bernoulli and multinomial Naïve Bayes are equivalent in principle. But in later experiments, the prediction accuracies of these 2 classifier in binary cases are slight different.

Algorithm Selection and Implementation

In the first part of this project, one Naïve Bayes classifier is implemented to classify the abstracts with binary features. A Laplace smoothing parameter of 0.01 is introduced to enhance the algorithm performance. In the third part, we choose logistic regression, Bernoulli Naïve Bayes and Multinomial Naïve Bayes to do the training and validation. In these 2 parts, we split 70% of the original data as training set and the rest 30% as validation set.

Preprocessing section for multinomial and svm

Feature extraction package from scikit-learn is used for the feature selection process of multinomial Naïve Bayes and SVM algorithms. Class CountVectorizer is very convenient to split texts, extract features and tailor them to a certain amount defined by the user. The removal of punctuations and stop words were also done by this class. All algorithms in this project are trained and validated based on the bag-of-words model, but details vary for different algorithms. For generating prediction outputs which should be uploaded to Kaggle, we tried both binary and non-binary cases, which differ in the representation of word appearance in text instances. Binary features uses 1 to stand for appearance and 0 for absence, while the value of non-binary feature represents how many times does one certain word(the feature) appear in an instance. Also, a stemmer implemented from package Natural Language Toolkit (nltk) is attached to CountVectorizer to transfer the original data.

However due to a limitation of laptop memory, not all features and instances can be used to train multinomial Naïve Bayes and SVM algorithms coded by ourselves. We select 10000 features and use all 88636 instances to train and validate multinomial Naïve Bayes, while 50 features and 8000 instances for SVM.

Results

-Feature Selection

Following figures and tables show how validation accuracies change as the number of features in consideration changes.

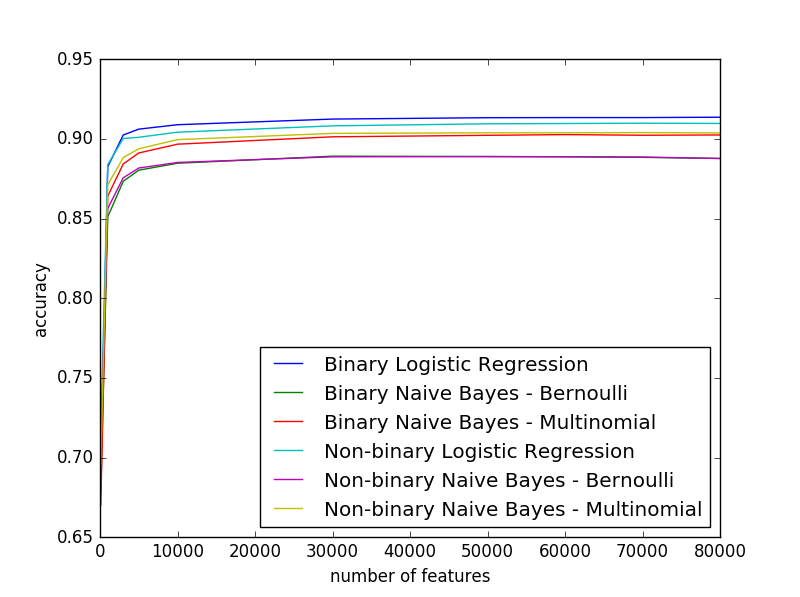


Fig. 1 Accuracy vs Number of Features Without Stemmer in Preprocessing

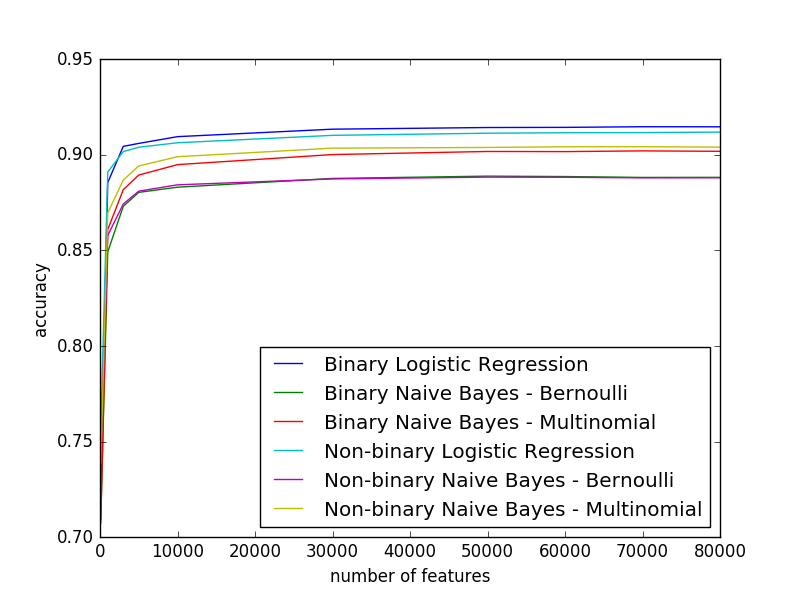


Fig. 2 Accuracy vs Number of Features with Stemmer in Preprocessing

Table 1 Validation Accuracy for Logistic Regression

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Num of features  Logistic Regression | Accuracy of binary feature | Accuracy of non- binary feature | Accuracy of binary feature with stemmer | Accuracy of non-binary feature with stemmer |
| 0.1k | 0.69452 | 0.71622 | 0.72700 | 0.75721 |
| 1k | 0.88255 | 0.88383 | 0.88553 | 0.89102 |
| 3k | 0.90245 | 0.90027 | 0.90444 | 0.90170 |
| 5k | 0.90613 | 0.90102 | 0.90598 | 0.90399 |
| 10k | 0.90892 | 0.90422 | 0.90952 | 0.90632 |
| 30k | 0.91245 | 0.90820 | 0.91343 | 0.91020 |
| 50k | 0.91332 | 0.90948 | 0.91429 | 0.91132 |
| 60k | 0.91339 | 0.90967 | 0.91437 | 0.91155 |
| 70k | 0.91339 | 0.90986 | 0.91471 | 0.91166 |
| 80k | 0.91362 | 0.90971 | 0.91467 | 0.91189 |

Table 2 Validation Accuracy for Bernoulli Naïve Bayes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Num of features  Bernoulli | Accuracy of binary feature | Accuracy of non- binary feature | Accuracy of binary feature with stemmer | Accuracy of non-binary feature with stemmer |
| 0.1k | 0.67867 | 0.70900 | 0.70960 | 0.74706 |
| 1k | 0.85104 | 0.85634 | 0.84923 | 0.85785 |
| 3k | 0.87342 | 0.87560 | 0.87308 | 0.87428 |
| 5k | 0.88037 | 0.88173 | 0.88037 | 0.88101 |
| 10k | 0.88474 | 0.88522 | 0.88308 | 0.88432 |
| 30k | 0.88917 | 0.88876 | 0.88767 | 0.88741 |
| 50k | 0.88895 | 0.88891 | 0.88887 | 0.88835 |
| 60k | 0.88876 | 0.88883 | 0.88865 | 0.88827 |
| 70k | 0.88842 | 0.88865 | 0.88820 | 0.88797 |
| 80k | 0.88774 | 0.88771 | 0.88823 | 0.88801 |

Table 3 Validation Accuracy for Multinomial Naïve Bayes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Num of features  Multinomial | Accuracy of binary feature | Accuracy of non- binary feature | Accuracy of binary feature with stemmer | Accuracy of non-binary feature with stemmer |
| 0.1k | 0.67011 | 0.69587 | 0.70738 | 0.73860 |
| 1k | 0.86409 | 0.87112 | 0.86093 | 0.86962 |
| 3k | 0.88428 | 0.88831 | 0.88176 | 0.88680 |
| 5k | 0.89117 | 0.89380 | 0.88944 | 0.89417 |
| 10k | 0.89669 | 0.89955 | 0.89489 | 0.89903 |
| 30k | 0.90132 | 0.90346 | 0.90015 | 0.90350 |
| 50k | 0.90226 | 0.90384 | 0.90177 | 0.90388 |
| 60k | 0.90271 | 0.90384 | 0.90170 | 0.90422 |
| 70k | 0.90230 | 0.90399 | 0.90207 | 0.90425 |
| 80k | 0.90249 | 0.90376 | 0.90185 | 0.90403 |

From the figures and table above, we can see that the predict accuracy not always increases with the number of selected features. For multinomial and Bernoulli Naïve Bayes, the accuracy reaches a peak when computed with approximately 70000 features. However the accuracies have already tended to be flat when the feature number exceeds 10000. The algorithms slightly improved when the feature set continue to grow. In consideration of memory consuming and computation duration, it is not necessary for us to take into account a large set of features. Also, the performance difference between binary and non-binary features is obvious when the feature set is relatively small and become negligible when the number of feature grows large. For the training and validation of classification algorithms from scikit-learn, 70000 non-binary features with all instances in train\_in.csv are considered to predict for the test set test\_in.csv. Also due to the limitation of laptop memory and computation time, 50 binary features and 8000 random selected instances are inputted to train and validate SVM. Since the stemmer from nltk always takes too long to process, our further algorithm experiments will not use stemmer to generate features.

-Multinomial Naïve Bayes

Figure 3 and Table 4 shows the prediction accuracies of Bernoulli and multinomial Naïve Bayes from scikit-learn as well as the algorithm implemented by our team. Only binary features are considered. Our Naïve Bayes classifier did a better job than the BernoulliNB but fell behind the MultinomialNB. However by observation the prediction stage of Naïve Bayes classifier by our team is significantly longer than the classifiers from package, which means further optimization may be needed. All the algorithms reach their accuracy peaks when the feature set approximately contains 50k to 60k words.

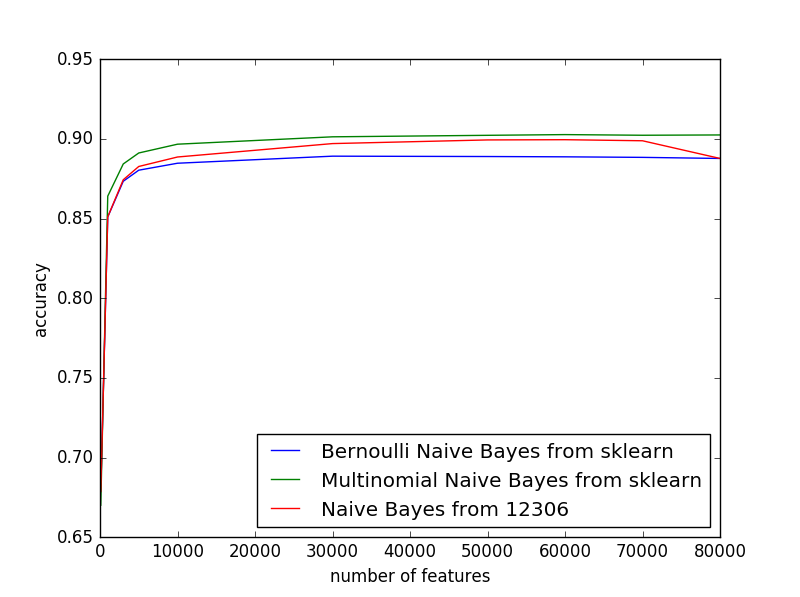


Fig. 3 Accuracy vs Number of Features with Naïve Bayes classifiers

Table 4 Validation Accuracy Comparison of Naïve Bayes from scikit-learn and Implemented by Our Team

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Features | Bernoulli from scikit-learn | Multinomial Naïve Bayes from scikit-learn | Naïve Bayes by 12306 |
| 0.1k | 0.67869 | 0.67011 | 0.67876 |
| 1k | 0.85104 | 0.86409 | 0.85108 |
| 3k | 0.87342 | 0.88428 | 0.87436 |
| 5k | 0.88037 | 0.89117 | 0.88270 |
| 10k | 0.88474 | 0.89669 | 0.88865 |
| 30k | 0.88917 | 0.90132 | 0.89707 |
| 50k | 0.88895 | 0.90226 | 0.89936 |
| 60k | 0.88876 | 0.90271 | 0.89951 |
| 70k | 0.88842 | 0.90230 | 0.89884 |
| 80k | 0.88774 | 0.90249 | 0.88774 |

Figure 4-6 illustrate the confusion matrix of the 3 Naïve Bayes classifers trained and validated with 60k features.

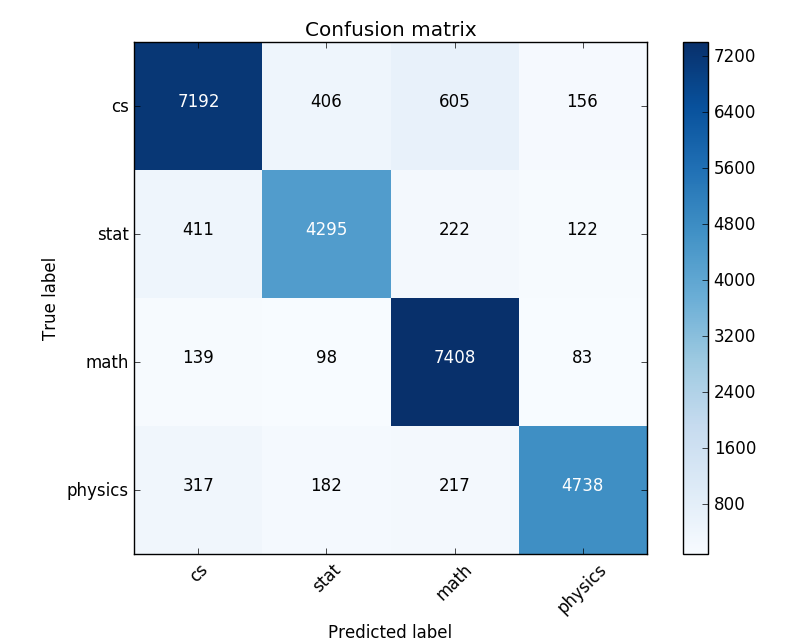


Fig 4 Confusion Matrix of Bernoulli Naïve Bayes

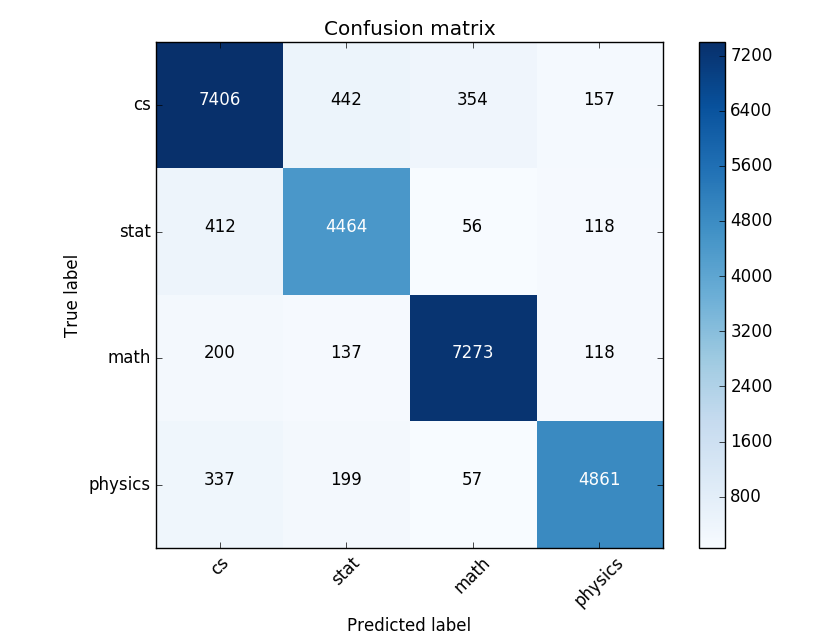


Fig 5 Confusion Matrix of Multinomial Naïve Bayes

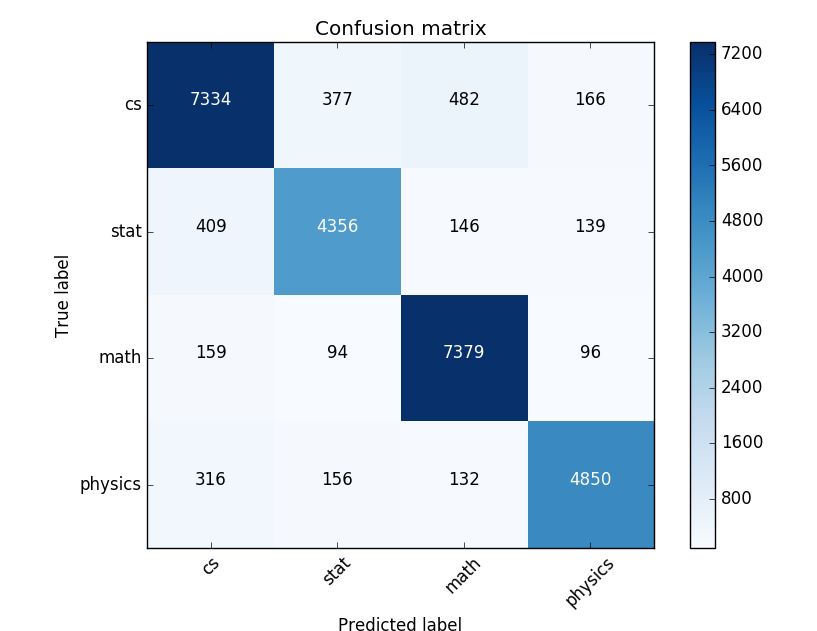


Fig 6 Confusion Matrix of Naïve Bayes Implemented by 12306

From the confusion matrices, we can see that Naïve Bayes can correctly classify most abstracts in categories ‘cs’ and ‘math’ while classification mistakes happen more often in the rest 2 categories, especially in stat. Among the misclassified samples in ‘stat’, around 60% are classified as ‘cs’, which may resulted from a number of terminologies shared between computer science and statistics.

Reference

[1] Metsis, Vangelis, Ion Androutsopoulos, and Georgios Paliouras. "Spam filtering with naive bayes-which naive bayes?." In CEAS, pp. 27-28. 2006.

[2] scikit-learn 0.18, <http://scikit-learn.org/>

[3] Natural Language Toolkit 3.0, http://www.nltk.org/