Assignment1: Text classification

***Abstract*—Text classification is the process of assigning documents (ie: web page, library book, media articles, gallery etc...) to one or more predefined categories. It has widely studied and has been one of the most typical task in supervised machine learning (ML). In this assignment, the documents extracted from Wikipedia were classified into two categories: "Artificial Intelligence (AI)" and "No-AI". The documents were pre-processed into bag-of-word and the features of each document were ranked by TF-IDF scores. Several classifiers were applied to classify the documents based on the extracted features. The classifiers performances were evaluated by the accuracy, precision, recall, F1 score of the validation files classification.**

I. Introduction

TEXT classification is a typical task which has been widely studied by [1], [2], [3], etc.. In these papers, several new approaches has been proposed to improve the performance of complicated text classification (ie: Long-Short Term Memory, Convolutional Nerual Network, etc.).

In this paper, we reported our several traditional approaches performances on a binary classification problem: classify the documents extracted from Wikipedia into "AI" and "No-AI" categories. The documents were pre-processed by firstly extracting the N-gram[4] and than ranked the N-gram by TF-IDF scores[5]. Eight ML models were selected to classify the documents, which are Naive Bayes[6], Stochastic Gradient Descent (SGD), Linear Support Vector Classifier (SVC)[7], Nearest Centroid[8], K Nearest Neighbours (KNN)[9], Passive Aggressive[10], Perceptron[11], Ridge Regression[12] and Random Forest[13] classifiers. The classifiers performances were evaluated by the accuracy, precision, recall, F1 score of the validation files classification.

The reminder of the report is organized as follows: Section II describes the pre-processing of transforming documents into a suitable representation for classification task. The Random Forest algorithm and Naive Bayes are introduced in Section III. The evaluation of different algorithms are provided in Section IV with different aspects. Section V shows the accuracy and time cost of different algorithms. Section VI provides a brief conclusion.

II. Data Extraction and Pre-processing

In order to classify text, the categorized text needs to first extracted and characterized into the format which could be understood by the ML methods. In this section, the data extraction and pre-processing pipeline is reported. Text data was firstly extracted by corpus extraction which is described in subsection II-A, while the text data is then been pre-processed into characterized vector by bag of word, term frequency-inverse document frequency (tf-idf) and n-grams in subsection II-B.

A. Corpus extraction

In this assignment, 3000 documents in AI category from Wikipedia were extracted to be the positive examples, whereas 3000 documents in Biology category and 3000 in Computer Science (CS) category were extracted to be the negative examples. The three groups documents were splitted into two sets of datasets: One is AI and Biology dataset, the other is AI and CS dataset.

However, the raw documents which were directly extracted from Wikipedia had many noises (ie: HTML, suffix, references, formula or other tags etc.). These noise were removed when extracting the data using mediaWiki API to improve the classification accuracy. Besides, the common words in the contents do not contain important information, those word are called stopwords. The stopwords were also removed based on the stopwords.txt file. All the cleaned files were stored in TXT file format.

B. N-gram extraction

After removing the noise from contents, ngrams (consecutive sequences of n characters) were extracted from long text strings. In this assignment, we used different length of ngrams, the length was from one character to five characters. The ngrams were ranked by computing their tf-idf score. TF-IDF is used as a weighting factor to measure how important a word is to a document in a collection or corpus. Then tfidf is calculated as Equation 1.

$$\begin{cases} \mathrm{tf}(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d}:t' \in d\}} \\ \mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \\ \mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D) \end{cases} \tag{1}$$

where N is the total number of documents in the corpus, $|\{d \in D : t \in d\}|$ is the number of documents where the term appears. If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $|1 + |\{d \in D : t \in d\}|$. The ngram which has higher value of tf-idf ranks higher than the ngrams with lower tf-idf value.

The feature selection was used to select the top k number of ngrams. The selected features are vectorized into a bag of word model. The bag-of-words model is used to convert the document into a feature for training a classifier. In this model, all the selected ngrams were represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. In this case, the collection of top-k words are saved into a sparse feature vector, while the value in the sparse features is the tf-idf value in the corresponding documents. Therefore, the classifiers could categorize the documents based on their bag-of-words sparse vector.

III. Description of algorithms

Eight classifiers were selected to solve the classification problem. In this section, we tried to simply describe these algorithms. Two Naive Bayes algorithms are described in subsection III-A and the other algorithms are briefly described in subsection III-B

A. Naive Bayes

Naive Bayes is one of the prevalent and standard algorithm in text classification and information retrieval area. In this assignment, two different kinds of models based on the original Bayes rule were selected. One is multi-variate Bernoulli model, the other is the multi-nomial model. In the next subsections, the classification based on the original Bayes' rule and the models of text classification will be introduced. The derivation of the two models will be discussed in Section IV.

1) Bayes' Rule: Naive Bayes classifier is based on Bayes rule:

$$P\left(c_j|d_i;\hat{\theta}\right) = \frac{P\left(c_j|\hat{\theta}\right)P\left(d_i|c_j;\hat{\theta}_j\right)}{P\left(d_i|\hat{\theta}\right)} \tag{2}$$

where, $P\left(c_j|d_i;\hat{\theta}\right)$ refers to the posterior probability of class c_j given the evidence of the test document d_i, $P\left(c_j|\hat{\theta}\right)$ refers to the the class c_j prior probability, $P\left(d_i|c_j;\hat{\theta}_j\right)$ refers to the probability of a document d_i given its class c_j, $P\left(d_i|\hat{\theta}\right)$ refers to he likelihood of a document d_i.

Given the training dataset and the maximum probability equation, classification can be implemented by calculating the posterior probability using the formula(2) and choose the maximum one to label the class.

In formula(2), the $P\left(d_i|c_j;\hat{\theta}_j\right)$ distribution based on two models. One is multi-variate Bernoulli event model, the other is multinomial event model. Using a distribution based on a multi-variate Bernoulli event model is called Bernoulli Naive Bayes classifier, using a distribution based on multinomial event model is called Multi-nomial Naive Bayes classifier.

2) Bernoulli Naive Bayes: Bernoulli Naive Bayes classifier is an implementation based on Bernoulli distribution. In the multi-variate Bernoulli event model, a document is represented by a binary vector over the space of words. Based on the naive Bayes assumption that the occurrence of each word in a documentation is independent, the probability of a document is simply the product of the probability of the attribute values over all word attributes:

$$\begin{aligned} P\left(d_i|c_j;\theta_j\right) = \prod_{t=1}^{|v|} \left(B_{it}P\left(w_t|c_j;\theta\right) + \right. \\ \left. \left(1-B_{it}\right)\left(1-P\left(w_t|c_j;\theta\right)\right)\right) \end{aligned} \tag{3}$$

In the formula(3), $t\epsilon\left\{1,...,|v|\right\}$ represents a dimension of V which represents a vocabulary, each dimension t corresponds to a word w_t from the vocabulary, its value is either 0 or 1, indicating whether word w_t occurs at least once in the document d_i.

3) Multi-nomial Naive Bayes: Different from multi-variate Bernoulli event model, multinomial model counts the word frequency instead of whether it occurring in the document d_i. The multinomial model views the document as an ordered event sequence extracted from the same vocabulary. Assume the length of the document is independent from the class and a similar Bayes assumption can be: the probability of the word event is independent from the other word and its position in the document. This will make a bag of words model for every document. Let N_{it} to be the count of the frequency of word w_t occurs in document. Then the probability of a document d_i is:

$$P\left(d_i|c_j;\theta_j\right) = P\left(|d_i|\right)|d_i|!\prod_{t=1}^{|v|}\frac{P\left(w_t|c_j;\theta\right)^{N_{it}}}{N_{it}!} \tag{4}$$

Formula(3) indicates that Bernoulli Naive Bayes model does not capture the number of times N_{it} each word occurs, and that it explicitly includes the non-occurrence probability of words that do not appear in the document. Whereas formula(4) indicates that Multi-nomial Naive Bayes does not include the non-occurrence probability but focus on the frequency.

B. Other algorithms

In this subsection, other different classifiers were described:

- Random Forest
- Linear Support Vector Classification (LinearSVC);
- Stochastic Gradient Descent (SGD);
- K-nearest neighbors (kNN);
- Nearest centroid;
- Passive Aggressive (PA);
- Perceptron;
- Ridge regression.

Random forest algorithm is a supervised classification algorithm, it is an ensemble of a set of decision trees. The steps of random Forest construction is to randomly select a subspace of features at each node to grow branches of a decision trees, then use bagging method to generate training data subsets for building individual trees, finally combine all the individual trees to form random Forests model.

Linear Support Vector Classification (LinearSVC) is a implementation of Support Vector Machine (SVM)[14] classifier. SVM use a mechanism called kernels, which essentially calculate distance between two observations. The SVM algorithm then finds a decision boundary that maximizes the distance between the closest members of separate classes. In this assignment, we used L1, L2 norm in the penalization.

SGD is a simple very efficient approach to discriminative learning of linear classifiers under convex loss functions such as SVM and Logistic Regression. SGD has the advantages of being efficiency, however, it is very sensitive to feature scaling. In this assignment, we used L1, L2, L1+L2 norm in the penalization.

kNN classifies the document by a majority vote of its neighbors, with the document being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the document is simply assigned to the class of that single nearest neighbor.

Nearest centroid classifier is a classification model that assigns to observations the label of the class of training samples whose mean (centroid) is closest to the observation.

PA classifier uses PA algorithms, where the loss function is the sum of a margin based term and a constant depending on the mistake type. In passive, if it is a correct classification, keep the model. In aggressive, if incorrect classification, update to adjust to this misclassified example.

The Perceptron classifier can be seen as the simplest type of artificial neural network. It is a model of a single neuron that can be used for two-class classification problems and provides the foundation for later developing much larger networks

Ridge Classifier is a classifier using Ridge regression. Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity. When multicollinearity occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value.

IV. Evaluation

In this section, we reported our experiment result in two datasets: AI and Biology dataset and AI and CS dataset. Each dataset is splited into train and validation set based on 10-fold cross-validation. We applies 8 ML classifiers to classify the documents based on the train set bag-of-word sparse vector and evaluates these classifiers performance in classification accuracy, time cost, precision, recall and F1 score. Besides, we also compare the classifiers performance in different feature selection strategies. The general performance of different algorithms and several feature selection strategies is reported in the following subsections.

All the experiment is implemented with python and sklearn. The code is runned in a 2.5 GHz Intel Core i7 with MacOS systems.

A. Algorithms Comparison

In this subsections, 8 ML classifiers is compared with each other in AI and Biology dataset and AI and CS dataset. The classification accuracy, time cost, precision, recall and F1 score are used to evaluate the classifiers.

In order to clarify the meaning of accuracy, precision, recall, f1-score, the confusion metrics is introduced. A confusion matrix is a table that is often used to describe the performance of a classification model. An example of confusion matrix in our cases is shown in Table I. In Table I, 4 terms are introduced True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN). TP and TN are the correctly predicted positive values and negative values, which means that the value of actual class is same with the the value of predicted class. FP is the case that the actual class is no-AI and predicted class is AI, while FN is the case that actual class is AI but predicted class in no-AI.

The calculation of accuracy, precision, recall, f1-score are derivative from TP, TN, FP and FN which is shown in Equation 5.

TABLE I
Example of Confusion Metrics

		Predict Class	
		Class = AI	Class = No-AI
Actual Class	Class = AI	True Positives	False Negatives
	Class = No-AI	False Positives	True Negatives

TABLE II
Results in AI and Biology dataset

	Accuracy	Precision	Recall	F1
Ridge	0.931	0.94	0.93	0.93
Perceptron	0.935	0.94	0.94	0.93
passive-agressive	0.945	0.95	0.94	0.94
KNN	0.945	0.95	0.94	0.94
Random Forest	0.945	0.95	0.94	0.94
Linear-SVC(L1)	0.939	0.94	0.94	0.94
Linear-SVC(L2)	0.935	0.94	0.94	0.94
SGDC(L1)	0.905	0.91	0.91	0.90
SGDC(L2)	0.912	0.92	0.91	0.91
SGDC(Elastic-net)	0.911	0.92	0.91	0.91
nearest centroid	0.895	0.90	0.90	0.90
naive-bayes(M)	0.906	0.91	0.91	0.91
naive-bayes(B)	0.909	0.91	0.91	0.91

$$\begin{cases} \text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN} \\ \text{Precision} = \frac{TP}{TP+FP} \\ \text{Recall} = \frac{TP}{TP+FN} \\ \text{F1} - \text{Score} = 2\frac{Recall \times Precision}{Recall+Precision} \end{cases} \tag{5}$$

Precision rate and recall rate are most commonly used in dichotomy questions. Precision rate is that how many true positive samples are in the samples that are labled as positive samples. And the recall rate is how many positive samples are labeled as positive. The accuracy measures the proportion of classification correctness. And the F1 score is the harmonic value of precision and recall value. By introducing the f1 score, the extreme situation where the precision or recall value is 0 can be avoid.

The accuracy, precision, recall, f1-score of different algorithms in AI and Biology is shown in Table II. The threshold of result is selected based on Zero-R. In our case, since the number of documents in AI and Biology is same, the threshold is equaled to 1/2. Besides, in this experiment the top-1000 key grams are selected to train and validate dataset. The best result in this dataset is obtained from passive-agressive, KNN and Random Forest, which have achieved 94.5% accuracy, 95% precision, 94% recall. The F1-score of these classifier is 0.94. The worst result is generated by nearest centroid, where its accuracy is 89.5%. The basic naive-bayes has achieved 90.9% accuracy.

Besides, the training and test time of different classifiers is shown in Fig. 2. Fig. 2a shows the classification accuracy and time cost of different classifiers between AI class and Biology class. Fig. 2b shows the accuracy and time cost of different classifiers between AI and Computer Science class. Random Forest had the highest training time, KNN had the highest test time. In our assignment, the time cost for Random Forest is 207.803s, whereas Naive Bayes only takes 0.075s during training the classifier to classify AI and Biology using 1,000 features. KNN took 208.273s to test the

test data category. Whereas the accuracy for Random Forest is 94.7% and 78.6% for Naive Bayes classifier.. In this case, Random Forest algorithm has the highest accuracy at the cost of highest training time. This is because its algorithm takes different features to create different decision trees and make a decision from those decision trees, the process of generating decision trees can improve accuracy by avoid overfitting, but also causes a huge memory consumption and training time.

B. Feacture Selection

Feature selection was used to filter irrelevant or redundant features from features based on the tf-idf value. In our assignment, we chose the different number of features and the classification accuracy of the categories AI and Biology when we selected different size of features was shown in Fig. 1. In Fig. 1, the x axis represents the size of the features, while the y axis shows the accuracy of the different classifiers.

As the result shows in Fig. 1, the accuracy for Random Forest, KNN, Bernoulli Naive Bayes, Multi-nomial Naive Bayes are 86%, 86%, 78%, 77% when selected just 10 features in each document, whereas with the increase of the size of features, the classification accuracy shows a growing trend, and the accuracy reached 94%, 94%, 88%, 86% when selected 10,000 features. But at the same time the training time also increased with the size of features increasing. This is because when the number of features is small, the models are underfitting the traianing dataset, after increasing the number of features, new features allows the models to expand the hypothesis space and can allow the models to discover signals that improve the fit.

Also, AI, Biology documents have two kinds of relatively different features, the distance between the features of these two categories are relatively high, so even we just selected the top 10 features, the accuracy of different models can also outperform the Zero-R method which has an accuracy of 50%.

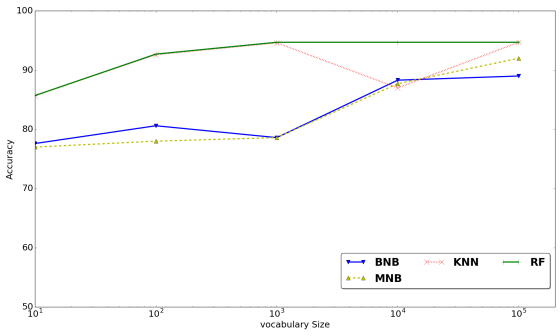


Fig. 1. TopK feature selection.

V. Conclusion

In this report, we first introduced how to pre-process the data for the text classification and how to split the train and validation dataset. After that, we report the 8 different classifiers performance in two dataset and compare the performance result with different feature selection strategies. Different evaluation metrics are applied to the evaluate the classifiers. Through finishing all the experiments, we have a intuitively understanding for the data preparation, feature extraction, evaluation metrics and classifiers. In the future, we may want to draw the precision and recall curve for the different classifiers.

References

[1] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[2] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification." in *AAAI*, vol. 333, 2015, pp. 2267–2273.

[3] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[4] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[6] A. McCallum, K. Nigam *et al.*, "A comparison of event models for naive bayes text classification," in *AAAI-98 workshop on learning for text categorization*, vol. 752, no. 1. Citeseer, 1998, pp. 41–48.

[7] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to platt's smo algorithm for svm classifier design," *Neural computation*, vol. 13, no. 3, pp. 637–649, 2001.

[8] I. Levner, "Feature selection and nearest centroid classification for protein mass spectrometry," *BMC bioinformatics*, vol. 6, no. 1, p. 68, 2005.

[9] S. Tan, "An effective refinement strategy for knn text classifier," *Expert Systems with Applications*, vol. 30, no. 2, pp. 290–298, 2006.

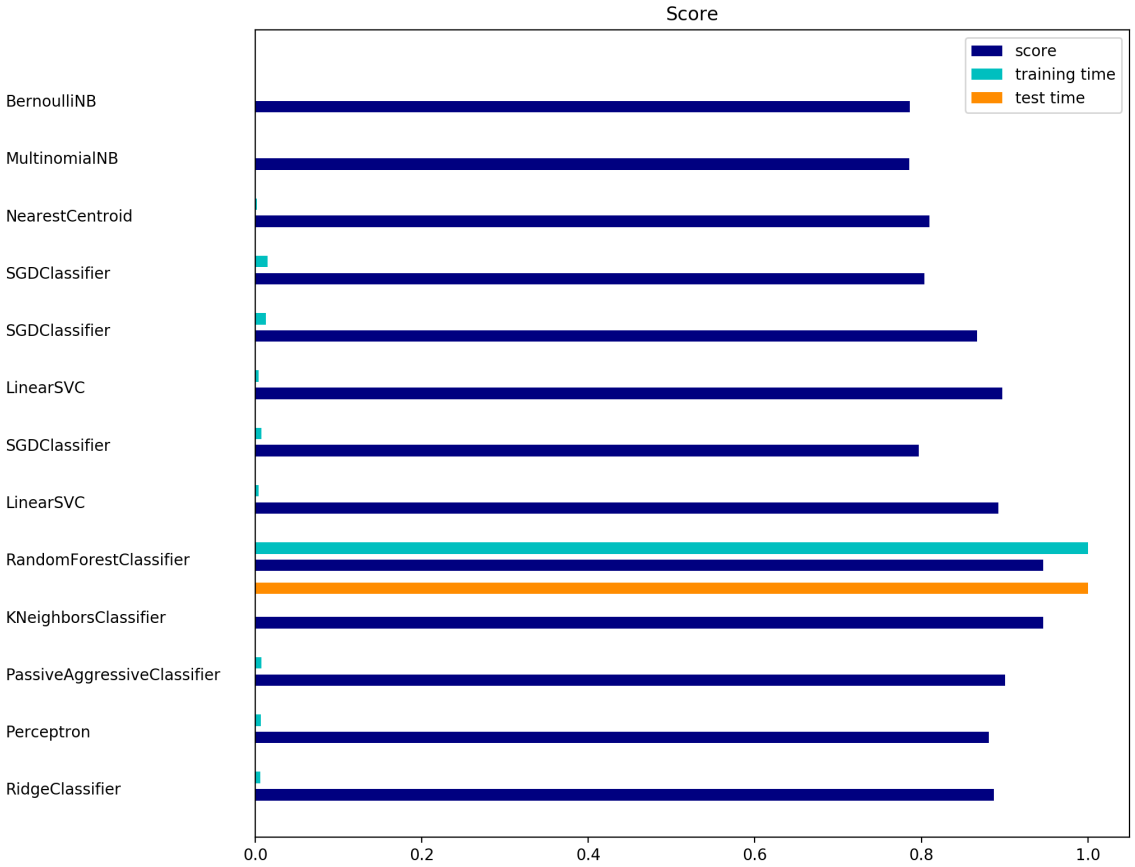
[10] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *Journal of Machine Learning Research*, vol. 7, no. Mar, pp. 551–585, 2006.

[11] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

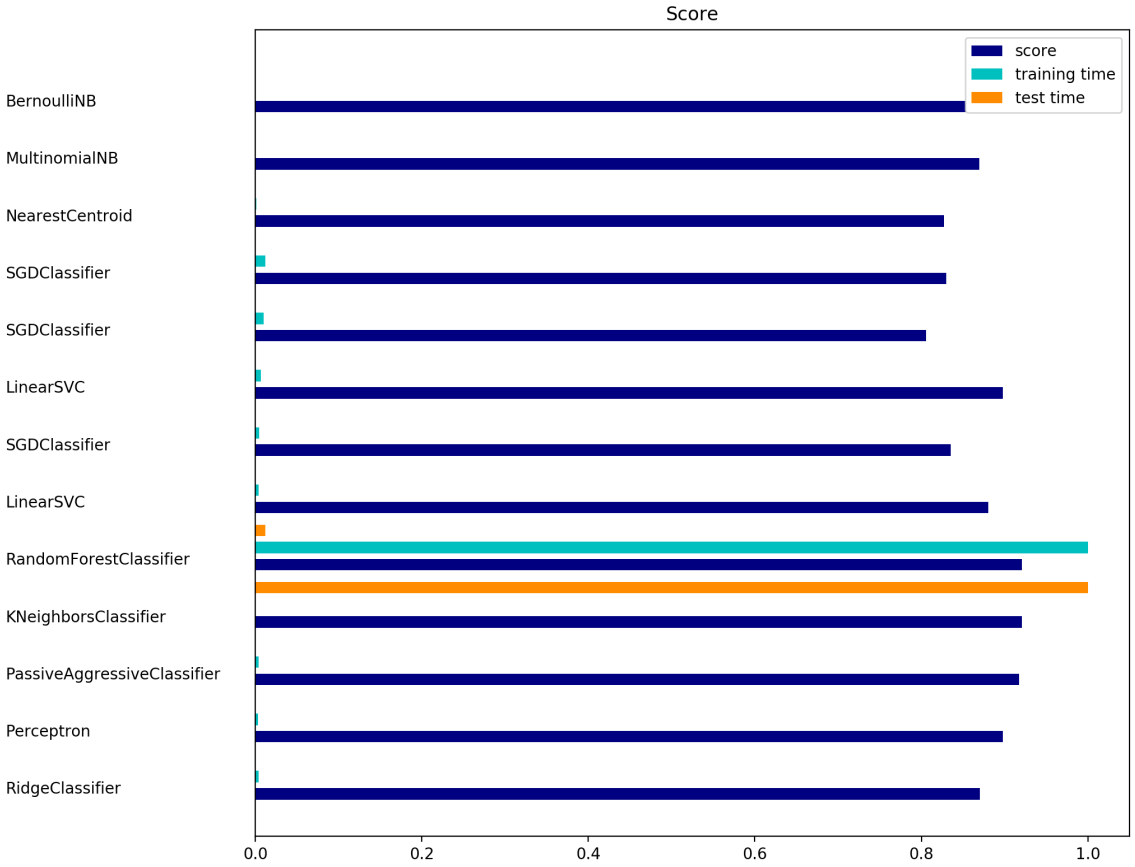
[12] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[13] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[14] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.



(a)



(b)

Fig. 2. Classification accuracy between two categories based on different algorithms. (a) AI and Biology classification results using Top1000 feature selection. (b) AI and Computer Science classification results using Top1000 feature selection.