Volume 6, Issue 1, January 2016 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

A Comparative Study of Classification Algorithms: k-Folds and Holdout as Accuracy Estimation Methods

Debby Erce Sondakh
Faculty of Computer Science, Universitas Klabat,
Indonesia

Abstract—This paper focuses on a comparative evaluation of two types of classifier accuracy estimation methods, k-folds cross validation and holdout, on five widely used classification algorithms. The algorithms considered here are decision tree, decision rules, Naïve Bayes, k-Nearest Neighbor, and Support Vector Machine. 10-folds cross validation and 70/30 percentage split were applied on the experiment. The algorithms are tested on multi-class text datasets. The results indicate that 10-folds cross validation method is superior to holdout method, as shown by the resulting accuracy, precision, and recall parameters. On the other hand, time spent for building the classifier by holdout method is shorter, compare to cross validation one. Among the five algorithms, naïve Bayes, Decision Tree, and Support Vector Machine outperform the other two. Naïve Bayes got the highest in all parameter tested, with a slightly slower classifier building time compare to k-NN.

Keywords—text classification, accuracy estimation, k-folds cross validation, holdouts

I. INTRODUCTION

The increase of digital documents available on the Internet becomes a trigger for a robust system that can organize the document and provide easy access to them. Data mining is widely used to handle the rapid growth of data, include digital documents. Documents are grouping into classes to simplify the information retrieval task from a large set of documents [1]. Two main approaches are applied, i.e. classification and clustering. Grouping process in classification technique relies on documents' predefined labels provided by experts. On the other hand, clustering group the documents based on documents' similarity.

Document classification is a two-step process, as depicted at Fig. 1. In the first step, called learning or training phase, a model is built by using classification algorithm, to analyze a predetermined set of training documents.

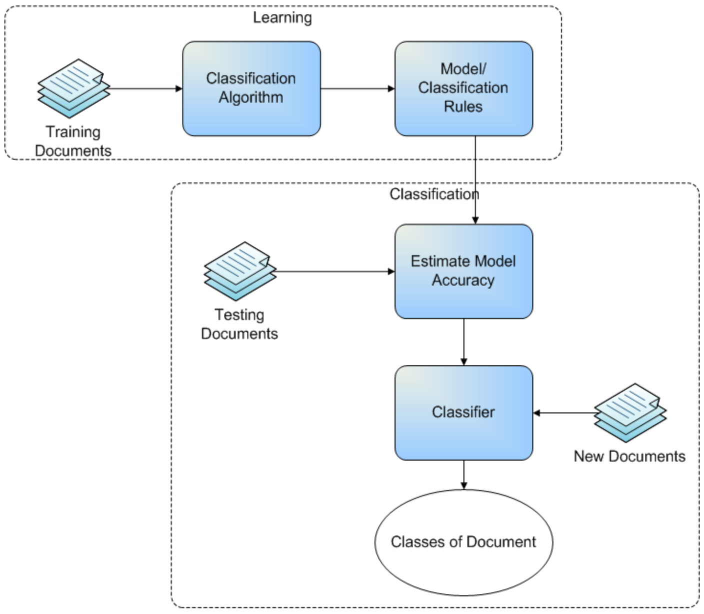

Fig. 1Classification Process

Each document is assumed to belong to a predefined class as determined by the class label attribute. The resulting classifier or model consists of classification rules describing the classifier's concept. In the second step, called classification, the resulting model from learning phase is evaluated to estimate its accuracy. Set of testing documents is used. This is independent from the training documents. Classifier accuracy is the percentage of test documents that are correctly classified [2]. The classifier can be used to classify new document of unknown class label only if its accuracy evaluation is considered acceptable (satisfy the minimum threshold).

Reference [3] and [4] define classifier accuracy as effectiveness, i.e. the classifiers' ability to classify a document in the right category. The effectiveness is measured in terms of precision, recall, and accuracy, based on probability technique. Holdout, random subsampling, cross validation (k-fold), and bootstrap are common techniques used for assessing classifier accuracy [2]. Holdout method partitions the full set of data into two sets, namely training set and test set. It is common to hold out two-third of the data for training (learning phase) and the remaining one-third of the data are for training (classification phase) [5]. Each set must be chosen independently and randomly. k-fold Cross Validation method randomly partitions the documents into k mutually exclusive subsets (called fold) $S_1, S_2, \dots, S_k$, each of approximately equal size. Model training and testing is performed $k$times, iteratively. At the i^{th}, subset S_i is reserved as the test documents while the remaining subsets are used as training documents. For example, at first iteration S_i will be the testing documents and $S_2, S_3, \dots, S_k$will be the training documents. Accuracy equals to the ratio of total appropriate classification from k iterations with total number of documents.

Which of these techniques can provide more accurate classifier? Which algorithm is more suitable for multi-class text classification? These two questions are addressed in this paper.

This paper is divided into five sections in addition to the Introduction. Section 2 introduces the related works to this research. Section 3 describes the five algorithms to be compared. The next section introduces two accuracy estimation methods applied in text document classification. Section 5 describes the datasets used, reports the simulation results and evaluation. The conclusion is summarized in section 6.

II. RELATED WORKS

This section presents five studies on comparing the accuracy of different classification algorithms for text documents. Ramasundaram& Victor [6] did a comparative study on six algorithms i.e. Naïve Bayes, SVM, N-Grams, k-NN, back propagation network and Genetic Algorithms. The study compared the classifier precision and execution time for twenty documents of various English dailies published in India. The categories for classification include politics, movies, sports, and education. The result shows that N-Grams algorithm has a minimum execution time, otherwise k-NN algorithm has maximum one. N-Gram's consistently produces lesser time than the others. It is closely followed by genetic algorithm and back propagation network. Comparison on the basis of precision also represents the same results. N-Gram has more precision on most documents. SVM and Naïve Bayes produce more precision on few documents.

Li & Jain [7] investigate the naïve Bayes, nearest neighbor, decision tree, and subspace method for document classification. These four different methods were applied to seven-class Yahoo news groups, individually and in combination. The experimental results indicate that naïve Bayes classifier and subspace method outperform the other two classifiers on individual evaluation, with accuracy 83.1% and 82.1% respectively.

In [8] comparison of SVM to k-nearest neighbor and naïve Bayes on binary classification tasks was studied. The aim of this research is to examine the classifier learning abilities for an increasing number of documents in the train set and how the number of attributes of the feature space affects the performance. The results show compared to SVM both k-nearest neighbor and naïve Bayes classifiers achieved good overall performance and are much faster than SVM. Naïve Bayes has an advantage when a small number of documents are used in the train set, but then as the number of document increases, the difference diminishes. The k-nearest neighbor continues to achieve very good result and scales up well with the number of documents, which is not the case for SVM. Processing time is a clear drawback for SVM. It tends to grow quadratically with the number of documents in the train set.

Wahbeh& Al-Kabi [9] conducted a comparative assessment of SVM, Naïve Bayes, and decision tree on Arabic text documents. They implemented percentage split and cross validation as the assessment method. In percentage split method the data is divided into 60% and 40% used for training and testing phase respectively. As for the cross validation method, researcher applied 10 fold cross validation. Experiments show that the naïve Bayes achieve the highest accuracy for both percentage split and cross validation methods, followed by SVM, followed by decision tree. In terms of time taken to build the models, SVM model is the lowest time, followed by naïve Bayes, and decision tree which takes a highest amount of time to build the model.

Ron [10] evaluated the cross validation and bootstrap accuracy estimation methods for decision tree and naïve Bayes algorithms, based on their resulting classifiers' variance and bias. The number of folds and bootstrap sample used are vary. [10] applied stratified and regular cross validation method.The result indicates stratified cross validation provides better result in terms of bias and variance, when compared to regular cross validation. Bootstrap method produced low variance, but extremely large bias.

III. DOCUMENT CLASSIFICATION ALGORITHMS

Document classification groups the documents into one or more categories based on their predefined label. The process is starting with the learning phase to determine the category of the document, is called supervised learning. This research investigated the text documents. Text classification is a relation between two sets, set of documents, $d = (d_1, d_2, \cdots, d_n)$ and set of categories $c = (c_1, c_2, \cdots, c_m)$. Document d_i will be classified into category c_j. $n$is the number of documents to be classified, and m is the total of predefined category in c [3, 4].

Text classification giving a Boolean value for each pair $(d_j, c_i) \in D \times C$, where D is the set of documents and C is a set of predefined categories. Classification is about to approximate the classifier function (also called rule, hypothesis, or model):

$$f: D \times C \rightarrow \{T, F\}$$

The value T (true) assigned to pair (d_j, c_i) indicates that document d_j includes in category c_i. Otherwise, the value F indicates that document d_j is not a member of category c_i [11].

In information retrieval, document is a sequence of words [5] that is stored as set of words called vocabulary or feature set [6]. Document is represent as Vector Space Model, i.e. an array of words, in the form of binary vector with value of 1 when a word present in the document or value of 0 for absences of a word. Each document is included in the vector space $R^{|V|}$,$|V|$ is the size of vocabularies V. For a collection of documents, called dataset, documents are represented as$m \times n$ matrix, where m is the number of documents and n is the words. Matrix elementa_{ij} denotes the occurrence of word j in document i which is represented as binary value.

A. Decision Rules

Decision rules using DNF rule to build a classifier for category c_i. DNF rule is a conditional rule consists of disjunctive-conjunctive clause; the rule describes the requirements for the document to be classified into categories defined; 'if and only if' the document meets on of the criteria in DNF clauses. They represent the categories' profile. Each single rule comprise of category's name and the 'dictionary' (list of words included in that category). A collection of rules is the union of some single rule using logic operator "OR". Decision rules will choose the rules whose scope is able to classify all the documents in training sets. Rules set can be simplified using heuristic without affecting the accuracy of resulting classifier.

Sebastiani in [3] explained, DNF rules are built in a bottom-up fashion, as follows:

1. Each training document d_j is $\eta_1, \ldots, \eta_n \rightarrow \gamma_i$ clause where $\eta_1, \ldots, \eta_n$ are the words contain in document d_j, and γ_i is the category c_i when d_j satisfy the criteria of c_i, otherwise it is $\overline{c_i}$.
2. Rules generalization. Simplifying the rules by removing the premise from clauses, or merging clauses. Compactness of the rules is maximized while at the same time not affecting the 'scope' property of the classifier.
3. Pruning. The resulting DNF rules from step 1 may contain more than one DNF clauses, which able to classify documents in the same category (overfit). Pruning is done to 'cut' the unused clauses from the rule.

B. Decision Tree

Decision tree decomposes the data space into a hierarchical structure called tree. In textual data context, data space describes the presence or absence of a word in the document. Decision tree classifier is a tree comprise of:

a. Internal nodes. Each internal node stores the attributes, i.e. collection of words, which will be compared with the words contained in a document.
b. Edge. Branches that come out of an internal node are the terms/conditions represent one attribute value.
c. Leaf. Leaf node is a category or class of documents.

Decision tree classifying document d_j by testing term weight of the internal nodes label contained in vector $\overline{d_j}$ recursively, until the document is classified at a leaf node. Label of the leaf node will be the document's class. Decision tree classifiers are built in a top-down fashion [3]:

1. Starting from the root node, document d_j is tested whether it has the same label as the node's (category c_i or $\overline{c_i}$).
2. If the does not fit, select the k-th term (t_k), divide into classes of documents that have the same value as t_k. Create a separated sub-tree for those classes.
3. Repeat step 2 in each sub-tree until a leaf node is formed. Leaf node will contain the documents in category c_i.

Decision tree's structure is easy to understand and interpret. Documents are classified based on their logical structure. On the contrary, this algorithm requires a long time to do the classification manually. When misclassification at the higher level occurs, it will affect the level below, and the possibility of overfit is high.

Sebastiani [3] explains, to reduce overfitting, several nodes can be trimmed (pruning), by withholding some of the attributes that are not used to build the tree. These attributes determine whether a leaf node will be pruned or not. The next step is comparing the class distribution in used attributes versus unused attributes. If the class distribution of the training documents used to construct the decision tree is different from the class distribution of the class distribution of the training documents retained for pruning, then the nodes are overfit to training documents and can be pruned.

C. k-Nearest Neighbor

In machine learning field k-nearest neighbor (k-NN) algorithm belongs to lazy learner group. Lazy learners, also called example-based classifier [3] or proximity-based classifier [12], do the classification task by utilizing the same existing category labels on the training documents with labels on the test documents.

k-NN starts by searching or determining the number of k nearest neighbor of the documents to be classified. Input parameter k indicates the number of document level to be considered in calculating document (d_j) classification function, $CSV_i(d_j)$. A document is compared with the neighbor classes, to calculate their similarity. Document d_j will become member of category c_i if there are k training documents that are similar to d_j in category c_i. k-NN classification function is defined as follows:

$$CSV_i(d_j) = \sum_{d_z \in Tr_k(d_j)} RSV(d_j, d_z) \cdot [\![\Phi(d_z, c_i)]\!]$$

- $RSV(d_j, d_z)$is a measure of relationship between testing document d_j with training document d_z.
- $Tr_k(d_j)$is the set of k testing document d_z to maximize the function $RSV(d_j, d_z)$.

D. Naïve Bayes

Naïve Bayes is a kind of probabilistic classifier that utilize mixture model, a model that combine terms probability with category, to predict document category probability [6]. This approach define classification as the probability of document d_j, which is represented as term vector $d_j = \langle w_{1j}, \ldots, w_{|T|j} \rangle$, belongs to category c_i.

Document probability is calculated using the following equation:

$$P_{(c_i|\vec{d}_j)} = \frac{P(c_i)P(\vec{d}_j|c_i)}{P(\vec{d}_j)}$$

$P(\vec{d}_j)$is the probability of document d_j (randomly chosen), $P(c_i)$ is the probability of a document to become classified in category c_i.

The size of document vector $\vec{d}_j$ may be large. Therefore, naïve Bayes applies word independence assumption. According to word independence assumption two different document vector coordinates are disjoint [3]. In other words, a term probability in a document does not depend on others. So, the presence of a word has no affect on others, so called ‘naïve’. Probabilistic classifier naïve Bayes is expressed in the following equation:

$$P(\vec{d}_j|c_i) = \prod_{k=1}^{|T|} P(w_{kj}|c_i)$$

There are two commonly used naïve Bayes variants, namely Multivariate Bernoulli and Multinomial Model.

a. **Multivariate Bernoulli Model**. This model using the term occurrence in document as the document feature. Term occurrence is represent as binary value, 1 for term presences and 0 otherwise. Term occurrence frequency is not taken into account for document classification modeling.
b. **Multinomial Model**. As oppose to multivariate model, this model considers the term occurrence frequency. Document is defined as ‘bag of words’, along with term frequency of each word. Classification modeling is conducted based on these occurrence frequencies in the document. Multinomial model has better performance compare with the other naïve Bayes variants [13, 14].

E. Support Vector Machine

Similar to regression-based classification, SVM represents documents as vectors. This approach aims to find a boundary, called decision surface or decision hyperplane, which separates two groups of vectors/classes. The system was trained using positive and negative samples from each category, and then calculated boundary between those categories. Documents are classified by first calculating their vectors and partition the vector space to determine where the document vector is located. The best decision hyperplane is selected from a set of decision hyperplane$\sigma_1, \sigma_2, \ldots, \sigma_n$ in vector space $|T|$ dimension that separate the positive and negative training documents. The best decision hyperplane is the one with the widest margin [3, 15].

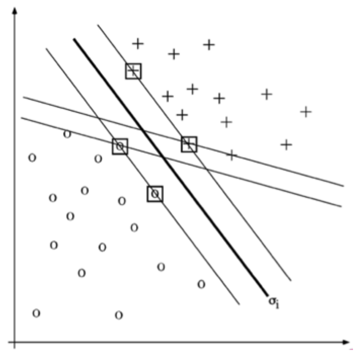

Fig. 2 Support Vector Classifier [3]

Fig. 2 shows how SVM work. The cross (+) and circle ( ) symbols represent two training document categories. Cross symbols for the positive ones and circle symbols otherwise. The lines represent decision hyperplanes, there are five decision hyperplanes on the example in Fig. 2. Box symbols are the support vectors, i.e. the documents whose distance against decision hyperplanes will be computed to determine the best hyperplane. $\sigma_i$is the best one. Its normal distance against each training documents is the widest. Thus,$\sigma_i$become the maximum possible separation barrier.

IV. RESULT AND EVALUATION

Based on these aspects: precision, recall, accuracy, and time taken to build the classifier, the performances of five classification algorithms were compare for seventeen text document datasets. The data used for the experiment is 17 multi-class text datasets taken from 19MclassTextWc text dataset, downloaded from http://weka.wikispaces.com/Datasets. The summary of documents used is shown in Table I.

TABLE I SUMMARY OF DOCUMENT SET

Dataset	**Number of Documents**	**Number of Attributes**
fbis	2463	2001
la1s	3204	13196
la2s	3075	12433
oh0	1003	3183
oh5	918	3013
oh10	1050	3239
oh15	913	3101
re0	1504	2887
re1	1657	3759
tr11	414	6430
tr12	313	5805
tr21	336	7903
tr23	204	5833
tr31	927	10129
tr41	878	7455
tr45	690	8262
wap	1560	8461

The experiment was conducted in two ways, i.e. k-fold cross validation and holdout (percentage split) methods. The number of folds evaluated for k-fold cross validation is 10-folds. In the holdout method data is divided into two partitions, 70% used for training phase and the remaining 30% for testing phase. Table II, III, and IV show the results for 10-folds cross validation and holdout method obtained over the five classifiers respectively for accuracy, precision, and recall. The resulting experiments indicate that cross validation may produce better classifier compare to holdout method. As summarized in Fig 3, 4, and 5, values resulted from cross validation is higher than holdout's.

TABLE II CLASSIFIER ACCURACY (IN PERCENT)

Datasets	**Decision Rules**		**Decision Tree**		**k-NN**		**Naïve Bayes**		**SVM**	
	10-folds	**70/30 holdout**	**10-folds**	**70/30 holdout**	**10-folds**	**70/30 holdout**	**10-folds**	**70/30 holdout**	**10-folds**	**70/30 holdout**
fbis	35.73	35.18	72.84	73.88	51.48	47.50	76.90	77.94	78.40	78.48
la1s	44.91	44.75	76.65	76.90	44.35	47.35	88.23	88.55	84.24	84.91
la2s	46.67	47.61	77.14	77.66	42.80	41.00	89.79	91.00	86.73	86.23
oh0	35.09	34.88	80.46	80.73	21.53	24.92	89.03	90.37	81.95	81.06
oh5	24.73	18.91	82.24	80.36	21.68	20	86.71	87.64	77.45	74.55
oh10	26.19	23.81	72.95	72.38	8.38	7.94	81.24	83.17	74.86	71.43
oh15	27.71	26.64	74.81	75.55	18.73	15.69	83.79	83.21	72.73	75.91
re0	57.45	53.66	75.27	74.72	62.17	61.64	80.39	78.49	75.47	73.84
re1	39.89	35.61	79.96	79.07	51.12	43.66	83.34	81.69	74.29	69.62
tr11	45.89	46.77	77.29	74.19	48.79	51.61	84.78	78.23	74.15	74.19
tr12	43.13	37.23	82.75	73.40	40.58	32.98	83.07	69.15	74.44	69.15
tr21	77.38	76.24	79.17	84.16	67.56	74.26	63.39	66.34	79.46	81.19
tr23	62.75	62.30	91.67	88.52	54.90	60.66	71.57	72.13	74.02	77.05
tr31	58.25	61.51	93.85	93.53	64.83	61.51	94.61	92.45	92.13	88.85
tr41	44.76	49.43	90.89	88.97	51.25	52.85	94.42	93.92	87.02	87
tr45	40.43	38.16	91.16	88.89	41.88	36.72	83.04	82.61	81.16	79.71
wap	30.19	25.85	65.64	66.67	39.87	33.55	81.09	79.70	82.18	79.70

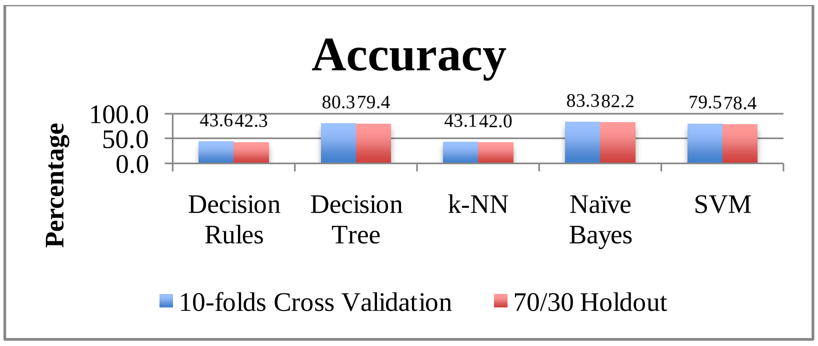

Fig. 3 Comparison of 10-folds Cross Validation and Holdout - Accuracy

In most cases, using either 10-folds cross validation and holdout methods, naïve Bayes outperform the other algorithms. The comparison of classifier on the basis of accuracy is depicted in Table II and Fig. 3. Naïve Bayes classifier achieves the highest accuracy (in average) 83.3%, using 10-folds cross validation. It is followed by decision tree (80.3%), then SVM (79.5%). The lowest accuracy is k-NN (43.1%). It can be seen in Fig. 3 that there are improvements to the accuracy of the five algorithms when applying cross validation method instead of holdout one. For both naïve Bayes and SVM methods are increased by 1.1% and decision tree's accuracy increases by 0.9%. As for decision rule and k-NN, the accuracy are increased by 1.3% and 1.1% respectively.

TABLE III CLASSIFIER PRECISION VALUE

Datasets	**Decision Rules**		**Decision Tree**		**k-NN**		**Naïve Bayes**		**SVM**	
	10-folds	**70/30 holdout**	**10-folds**	**70/30 holdout**	**10-folds**	**70/30 holdout**	**10-folds**	**70/30 holdout**	**10-folds**	**70/30 holdout**
fbis	0.357	0.135	0.725	0.74	0.586	0.526	0.794	0.8	0.781	0.785
la1s	0.449	0.294	0.764	0.766	0.595	0.559	0.881	0.886	0.844	0.85
la2s	0.467	0.356	0.772	0.777	0.58	0.561	0.898	0.91	0.868	0.864
oh0	0.351	0.246	0.821	0.836	0.505	0.339	0.893	0.904	0.826	0.818
oh5	0.247	0.081	0.834	0.81	0.494	0.523	0.869	0.885	0.783	0.748
oh10	0.262	0.132	0.74	0.727	0.375	0.315	0.808	0.835	0.745	0.738
oh15	0.277	0.163	0.761	0.777	0.612	0.693	0.839	0.842	0.732	0.777
re0	0.574	0.336	0.747	0.75	0.633	0.621	0.821	0.795	0.764	0.766
re1	0.399	0.166	0.793	0.793	0.579	0.534	0.83	0.799	0.758	0.744
tr11	0.459	0.255	0.767	0.748	0.551	0.59	0.842	0.783	0.741	0.716
tr12	0.431	0.186	0.823	0.708	0.518	0.553	0.845	0.742	0.768	0.786
tr21	0.774	0.598	0.778	0.84	0.669	0.682	0.816	0.817	0.789	0.804
tr23	0.627	0.507	0.917	0.906	0.611	0.592	0.844	0.794	0.737	0.755
tr31	0.583	0.68	0.937	0.934	0.665	0.68	0.946	0.925	0.92	0.885
tr41	0.448	0.305	0.909	0.903	0.676	0.637	0.946	0.928	0.876	0.861
tr45	0.404	0.245	0.916	0.904	0.482	0.507	0.852	0.851	0.809	0.81
wap	0.302	0.176	0.661	0.695	0.641	0.658	0.802	0.795	0.835	0.811

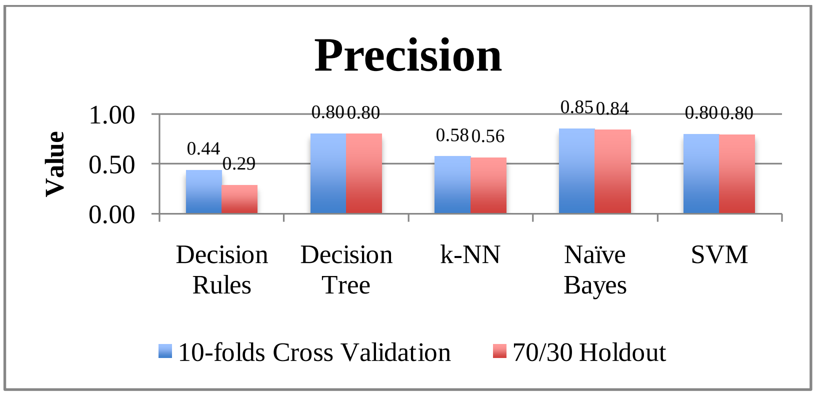

Fig. 4 Comparison of 10-folds Cross Validation and Holdout – Precision

The precision values, as shown in Table III and Fig. 4, represent the same type of result as Table II and Fig. 3. In average, naïve Bayes' precision value is still the highest i.e. 0.85 using cross validation and 0.84 for holdout method. It is closely followed by SVM and decision tree which have 0.8 for each one. The precision value of k-NN is better than its accuracy. It is 0.58 using cross validation, 0.02 point higher than holdout method. The decision tree algorithm, unlike the other four, its precision value resulting from cross validation method is lower than holdout. Fig. 4 shows the precision value decreases by 0.15 point.

TABLE IV CLASSIFIER RECALL VALUE

Datasets	**Decision Rules**		**Decision Tree**		**k-NN**		**Naïve Bayes**		**SVM**	
	10-folds	**70/30 holdout**	**10-folds**	**70/30 holdout**	**10-folds**	**70/30 holdout**	**10-folds**	**70/30 holdout**	**10-folds**	**70/30 holdout**
fbis	0.144	0.352	0.728	0.739	0.515	0.475	0.769	0.779	0.784	0.785
la1s	0.305	0.447	0.767	0.769	0.444	0.473	0.882	0.886	0.842	0.849
la2s	0.332	0.476	0.771	0.777	0.428	0.41	0.898	0.91	0.867	0.862
oh0	0.232	0.349	0.805	0.807	0.215	0.249	0.89	0.904	0.82	0.811
oh5	0.28	0.189	0.822	0.804	0.217	0.2	0.867	0.876	0.775	0.745
oh10	0.143	0.238	0.73	0.724	0.084	0.079	0.812	0.832	0.749	0.714

Datasets	**Decision Rules**		**Decision Tree**		**k-NN**		**Naïve Bayes**		**SVM**	
	10-folds	**70/30 holdout**	**10-folds**	**70/30 holdout**	**10-folds**	**70/30 holdout**	**10-folds**	**70/30 holdout**	**10-folds**	**70/30 holdout**
oh15	0.144	0.266	0.748	0.755	0.187	0.157	0.838	0.832	0.727	0.759
re0	0.375	0.537	0.753	0.747	0.622	0.616	0.804	0.785	0.755	0.738
re1	0.202	0.356	0.8	0.791	0.511	0.437	0.833	0.817	0.743	0.696
tr11	0.335	0.468	0.773	0.742	0.795	0.516	0.486	0.782	0.845	0.742
tr12	0.274	0.372	0.827	0.734	0.406	0.33	0.831	0.691	0.744	0.691
tr21	0.616	0.762	0.792	0.842	0.676	0.743	0.634	0.663	0.795	0.812
tr23	0.528	0.623	0.917	0.885	0.549	0.607	0.716	0.721	0.74	0.77
tr31	0.441	0.615	0.939	0.935	0.648	0.615	0.946	0.924	0.921	0.888
tr41	0.213	0.494	0.909	0.89	0.513	0.529	0.944	0.939	0.87	0.875
tr45	0.404	0.382	0.912	0.889	0.419	0.367	0.83	0.833	0.812	0.797
wap	0.172	0.259	0.656	0.667	0.399	0.335	0.811	0.797	0.822	0.797

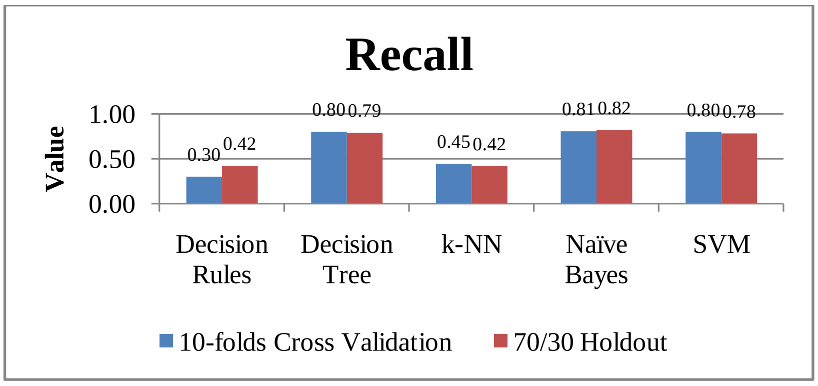

Fig. 5 Comparison of 10-folds Cross Validation and Holdout – Recall

In this research, comparison of the five classification algorithms was also assessed based on recall value, probability of the algorithm to retrieve the relevant document. Table IV contains the comparison of recall value resulting from the experiment, while Fig. 5 provides the average recall value. As depicted in Fig 5, decision tree and SVM achieve the same precision and recall value using cross validation method. But they decrease by 0.01 and 0.02 point, respectively, when holdout method applied. The Naïve Bayes classifier recall value slightly increases (0.01) when implementing holdout method. As for decision rules algorithm, there is an improvement when moving from the cross validation method to holdout method, where the recall value increases by 0.12 point.

TABLE V AVERAGE TIME TAKEN TO BUILD THE CLASSIFIER (IN SECOND)

Methods	**Decision Rules**	**Decision Tree**	**k-NN**	**Naïve Bayes**	**SVM**
10-folds Cross Validation	10.90	103.72	0.01	0.03	3.42
70/30 Holdout	10.15	99.63	0.00	0.02	3.73

Table V shows another measure that is obtained from the experiments, the amount of time taken to build the classifier. It is observed from Table V that, holdout method requires less time to build a classifier compare to k-fold cross validation, except for SVM algorithm. The k-NN classifier is fastest in terms of classifier building time for both evaluation methods, while the decision tree is the slowest. The other three algorithms that require a small amount of time to build the classifier are naïve Bayes, SVM, and decision rules. The decision tree, even it has good result for classifier effectiveness and accuracy, it suffers in classifier building time.

V. CONCLUSION

This study compared performance of five machine learning based classification algorithms, namely decision rules, decision tree, k-NN, naïve Bayes, and SVM. Comparison is based on classifier effectiveness measurements: precision, recall, accuracy, and time taken to build classifier. After the experiment and analysis of the results following conclusions were drawn:

1. Overall, k-folds cross validation shows better effectiveness and accuracy compare to holdout method.
2. Decision rules and k-NN performance are lack. Decision rules' effectiveness is the lowest, but better in accuracy compare to k-NN.
3. The Naïve Bayes, Decision Tree and SVM algorithms can build classifiers with high effectiveness rate and accuracy.

   a. SVM is able to classify the documents well in small amount of model building time.
   b. Decision tree has good performance in classifying multi-class text documents, with average precision and recall values more than 0.8, as well as accuracy rate which is more than 80%. Yet, it is weak in time to build the classifier models.
4. Experiment result shows naïve Bayes has the highest effectiveness values, as well as spent small amount of time to build the classifier models.

REFERENCES

[1] H. Brucher, G. Knolmayer, and M. A. Mittermayer, "Document Classification Methods for Organizing Explicit Knowledge", in *Proc. Of the 3rd European Conference on Organizational Knowledge, Learning, and Capabilities*, 2002.

[2] J. Han and M. Kamber, *Data Mining Concepts and Techniques*, USA: Academic Press, 2001.

[3] F. Sebastiani, "Machine Learning in Automated Text Categorization", *ACM Computing Surveys*, vol. 34, pp. 1-47, March 2002.

[4] M. Ikonomakis, S. Kotsiantis, and V. Tampakas, "*Text Classification Using Machine Learning Techniques*", *WEAS Transactions on Computer*, vol. 4, pp. 966-975, August 2005.

[5] I. H. Witten and E. Frank, *Data Mining Practical Machine Learning Tools and Techniques*, 2nd ed., Morgan Kaufmann Publishers, 2005.

[6] S. Ramasundaram and S.P. Victor, "*Algorithms for Text Categorization: A Comparative Study*", *World Applied Sciences Journal*, vol. 22, pp. 1232-1240, 2013.

[7] Y. H. Li and A. K. Jain, "*Classification of Text Document*", *The Computer Journal*, vol. 41, pp. 537-546, 1998.

[8] F. Colas and P. Brazdil, "*Comparison of SVM and Some Older Classification Algorithms in Text Classification Tasks*", ser. Artificial Intelligence in Theory and Practise, International Federation for Information Processing, M. Bramer,Ed., US: Springer, pp. 169-178, 2006, vol. 217.

[9] A. H. Wahbeh& M. Al-Kabi, "*Comparative Assessment of the Performance of Three WEKA Text Classifier Applied to Arabic Text*", *Basic Science & Engineering*, vol. 21, pp. 15-28, 2012.

[10] K. Ron, "*A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection*", in*Proc. Of Joint Conference on Artificial Intelligence 1995 (IJCAI 1995).*

[11] P. Y. Pawarand S. H. Gawande, "*A Comparative Study on Different Types of Approaches to Text Categorization*", *Int. Journal of Machine Learning and Computing*, vol. 2, August 2012.

[12] C. C. Aggarwaland C. X. Zhai, "*A Survey of Text Classification Algorithms*", *Mining Text Data,* Springer Science Business Media, 2012.

[13] A. Bratkoand B. Filipié, "*A Study of Approaches to Semi-structured Document Classification*", Josef Stefan Institute, Slovenia, Tech. Rep IJS-DP 9015, 2004.

[14] Y. Yang. and X. Liu, "*A Re-examination of Text Categorization Methods*", in*Proc. of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, pp. 42-49.

[15] E. Leopold and J. Kindermann, "*Text Categorization with Support Vector Machines. How to Represent Texts in Input space?*",*Machine Learning*, vol.46, pp. 423-444, 2002.