International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 12 (2015) pp. 31527-31540

http://www.ripublication.com

A New Supervised Learning Method and Its Application in Improving Classification Efficiency of Large Data Sets

Bhabani Shankar Das Mohapatra[1] and AmareAnagaw Ayele[2]

[1]Research Scholar, CSIT, JNT University, Hyderabad, India
dm_bhabani@yahoo.co.in
[2]Faculty of Computing, Institute of Technology,Bahir Dar University, Ethiopia
Livewithlee4ever@gmail.com

Abstract

Nowadays machine learning (ML) provides extremely important tools for data analysis, processing and visualization. Machine learning has become an important complement to the traditional techniques of data analysis. There are several applications for machine learning, the most significant of which is data mining. To organize the data, we need data mining concepts, which discover the knowledge by analyzing the large volumes of data from various dimensions and summarizing it into useful information. It is used to identify hidden patterns in a large dataset. Classification methods (supervised ML) are techniques that classify data items into predefined class label based on the training dataset. Classification method is one of the most useful techniques in data mining to build classification models from an input dataset.

In this study, we develop a novel supervised classification technique based on similarity matrix (*m*×*nSimMat*). Our performance study shows that *m*×*nSimMat*method is efficient and scalable for miningbased on supervised method and also better than some well-known classification techniques.

Keywords: *m*×*nSimMat*, k-NN, Naïve Bayesian, similarity matrix.

I. INTRODUCTION

Data mining, or knowledge discovery, is the computer assisted process of digging through and analyzing enormous sets of data and then extracting the meaningful information from the data. Data mining tools predict behaviors and future trends, allowing businesses to make proactive, knowledge driven decisions. Data mining tools can answer business questions that traditionally were tootime-consuming to

resolve. They scour databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations. Classification techniques, also known as supervised learning methods, are data mining techniquesin which we need a data set used to construct the model. The constructedmodel then tries to predict the class label of the unseen dataset. Classification consists of assigning a class label to a set of unclassified cases. The objective of classification is to analyze the input data and to develop an accurate description or model for each class using the features present in the data. This data called the *training set*, consists of multiple records each having multiple attributes or features, each record is tagged with a class label. Different classification algorithms have been developed, all are grouped either in the categories of eager learner (example naive) or lazy learner (example K-Nearest neighbor).Lazy learners construct a generalization model before receiving new tuples to classify. Learned models are ready and eager to classify previously unseen tuples.

In this article, we propose a new classification techniquethat belongs to lazy learner and exhibits better performance than the previous eager and lazy learners. We conducted the performance analysis of the proposed technique on the given UCI data sets called SPECT heart data (cardiac Single Proton Emission Computed Tomography data). The performance of the classification algorithm has been measured by accuracy, speed, robustness, scalability and interpretability. The proposed algorithm, called as $m \times nSimMat$,has shown high accuracy but poor in handling noise and missing value (robustness).

Classification that predicts categorical class labels (discrete or nominal), has many applications in the real world. Some of the applications are intrusion detection, market basket analysis, aids to marketing and retailing, medical and health care and banking and card risk scoring.All those applications need a novel classification algorithm. But all the existing eager and lazy algorithms have insufficient accuracy and their performance totally depend on the type of data in which they are applied. Even the performance of the algorithm cannotprovide cent percent accuracy even if they are tested on the training data.

Many classification techniques are used only for small datasets and require that all or a portion of the entire dataset remain permanently in memory. In practice, the value of k is usually optimized by many trials on the training and validation sets. Butthismethod isnot feasibleinsomecaseswhere wehave nochancetodo cross validation, such asonline classification[5].

For instance, one of the drawbacksofk-NN algorithm is its efficiency, as it needs to compare a test document with all samples in the training set. In addition, the performance of this algorithm greatly depends on two factors, that is, a suitable similarity function and an appropriate value for the parameter k [5].

Generally, aclassification algorithm has the following problem

- Has low performance (measured in accuracy), even if they are tested on the training data.
- The performance depends on the type of data the dataset contains.
- The performance depends on the number and type of tuples used as training set.

A number of data mining techniques such as Regression,Genetic algorithm,Baysclassification,k-meansclustering, associate rules, prediction etc. have already been implemented ondifferentdatatoimprovethe performanceofanalyzing the hidden knowledge that exists on the data. Data mining techniques can be used in different field to enhance our understanding of learning process to focus on identifying, extracting and evaluating variables related to the learning process of the world.Classificationisoneofthemostfrequently studiedproblemsbydataminingand machinelearning researchers(Brijesh et al,2011).Itconsistsofpredictingthevalueofacategorical attributebasedonthevalues ofotherattributes. There is a general architecture of all classification algorithms.

Classification methods like k-NN, rule mining, Bayesian network etc.canbeappliedonthedifferent dataforpredictingthe hidden knowledge among the data.In this paper, we try to compare our newly proposed method (*m*×*nSimMat*) with Naïve Bayesian and k-NN algorithms.

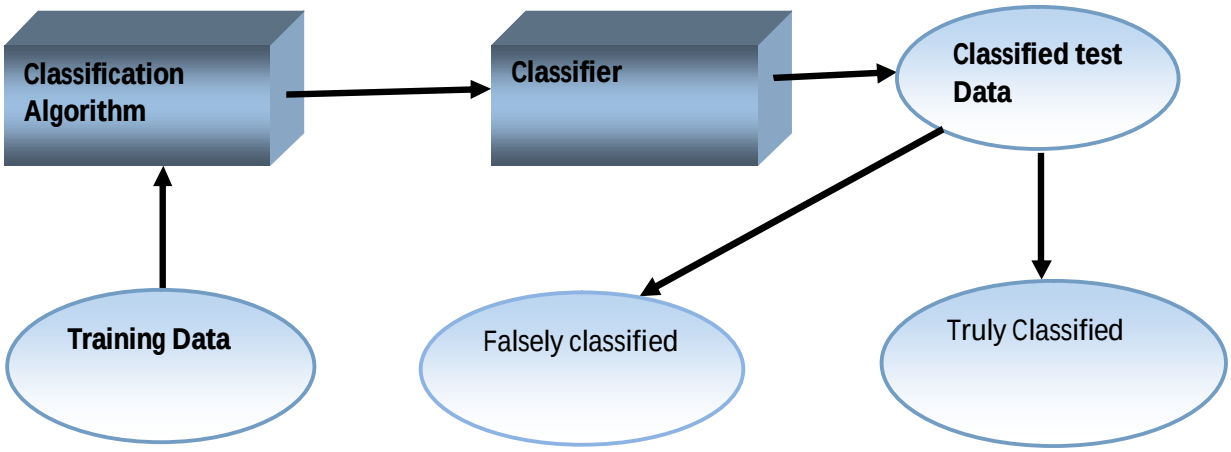

Fig.1: Architecture of general classification techniques

II. NAÏVE BAYESIAN

Naive Bayesian classification iscalled *naïve*, because it assumes class conditional independence. That is, the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is made to reduce computational costs, and hence is considered "naive". The major idea behind Naive Bayesian classification is to try and classify data by maximizing P(Xj|Ci)P(Ci) (where *i* is an index of the class) using the Bayes' theorem ofPosterior probability.

The naïve Bayesian classifier is one of the most popular data mining techniques for classifying the large dataset. Naive Bayesian classifier is a statistical classifier based on the Bayes' Theorem and the maximum posteriori hypothesis. The earliest description of Naive Bayesian classifier is found in [6]. Some of the reasons this classifier is so common and simple that it is easy to implement and fast since the naive assumption of class conditional independence reduces its computational cost [7].Experiential studies comparing classification algorithms have found that the naive

Bayesian classifier is comparable in performance with decision tree and selected neural network classifiers [8] [9].

Algorithm: Naïve Bayesian Classifier

Table 1: Algorithm of Naïve Bayesian

Input:
D: a training set of tuples and their associated class labels each tuple is represented by n-dimensional vector X(x_1,….,x_n), n measurements of n attributes A1,…,An
Output:
Classes: suppose there are m classes C_1,…, Cm
Method:
1) Given a tuple X, the classifier will predict that X belongs to the class having the class having the highest posterior probability conditioned on X.
2) Predict that tuple X belongs to the class Ci if and only if
$P(Ci|X)>P(Cj|X)$ for $1\leq j\leq m, j\neq i$
3) Maximize P(Ci|X): find the maximum posterior hypothesis.
$P(Ci|X)= P(X|Ci)P(Ci)/P(X)$
4) P(X) is constant for all classes, thus, maximize p (X|Ci)P(Ci).

III. K-NEAREST NEIGHBOR

k-Nearest Neighbor is one of the most popular algorithms for text categorization [1]. Many researchers have found that k-NN algorithm achieves very good performance in their experiments on different data sets [2][3][4]. Nearest-neighbor classifiers compare a given test tuple with training tuples that are similar training tuples described by n attributes. Training tuples are stored in n-dimensional space. The objective is to find the k-nearest tuples from the training set to the unknown tuple. The unknown tuple is assigned the most common class among its k-nearest neighbor. When k=1, the unknown tuple is assigned the class of the training tuple that is closest to it1-NN scheme that has a miss-classification probability that is no worse than twice that of the situation where we know the precise probability density of each function.

Algorithm*: *k-NN Classifier

Table 2: Algorithm of k-NN.

Input:

- *Let U be the unknown tuple whose class we want to assign*
- *Let T be the training set containing training tuple,* $T_1=(t_{1,1}, t_{1,2}, \ldots t_{1,n})$, $T_2=(t_{2,1}, t_{2,2},\ldots t_{2,n}),\ldots,T_m=(t_{m,1}, t_{m,2}\ldots t_{m,n})$.
- *Let attribute* $t_{i,n}$ *be the class label of* T_i
- *Let m be the number of training tuples.*
- *Let n be the number of attributes describing each tuple.*
- *Let k be the number of nearest neighbor we wish to find.*

Output: *Class label for U.*

Method: *the method is outlined as follow:*

1) Array a[m][2]://m rows containing data regarding the m training tuples. The first column is the Euclidean distance between U and that row's training tuple. The second column refers to that training tuple's index. We need to save the index because when sorting the array (according to Euclidean distance), we need some way to determine to which training set the Euclidean Distance refers.

2) For i=1 to m do{

3) a[i][1]=Euclidean distance(U, T_i*);*

4) a[i][2]=I;}// save the index, because rows will be sorted later

5) sort the rows of a by their Euclidean distance saved in a[i][1](in ascending order ;

6) array b[k][2];// the first column hold the distinct class labels of the k-nearest neighbor, while the second hold their respective counts. In the worst case, each k-nearest neighbor will have a different class label, which is why we need to allocate room for k class labels.

7) For i=1 to k do {

8) If class label ta[i][2], n already exist in array b then

9) Find that class label into the next available row of array b and increment its count;}

10) Else add the class label into the next available row of array b and increment its count);

11) Sort array b in descending order(from class label with largest count down to that with smallest count);

12) *Return (b [1]); //return most frequent class label of the k-nearest neighbor of U as the class prediction.*

If k=5, then in this case query instance $x_q$will beclassified as negative since three of its nearest neighbors are classified as negative as shown in the following figure.

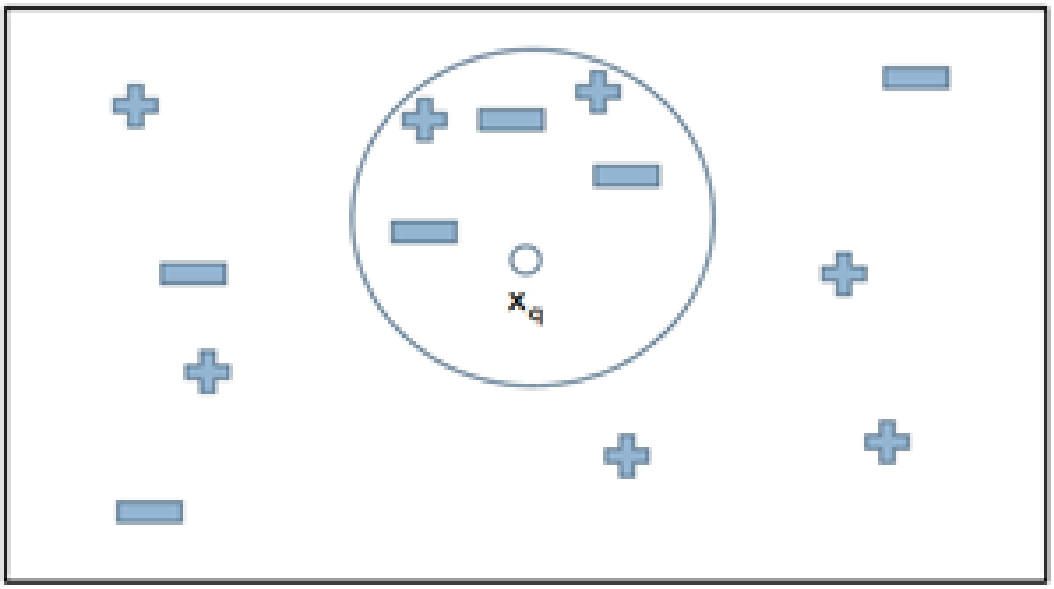

Fig. 2: K-nearest neighbor algorithm architecture

IV. THE PROPOSED ALGORITHM:(*m*×*nSimMat*)

The proposed algorithm is grouped to lazy learner group, in which they do less in the training and much when the testing tuple is presented. The algorithm has high performance if the possible values of the tuples are used as training dataset. The algorithm compares each value of the testing tuples with each value of the whole training tuples which are in the same column. If the values are identical, a '1' is placed and when it is not, a zero is placed in the multidimensional array *temp*[m][n], where m is the number of training tuples and n is the number of attributes columns of the test or training data. Then the algorithm adds the *temp* value row wise, and selects the highest value as a threshold value to determine how many of the training tuples have the same data as the test data. The algorithm then finds among them how many are in the same class label and concludes that the testing data has the same class label with that group. But this algorithm is sensitive to outlier and noisy data, if the number of *temp* values exceeding the threshold value are even numbers and if exactly half of them are in one class. The algorithm calls k-NN for that tuple to classify.

Analysis of the Algorithm:

The algorithm tries to find similarity matrix of the testing data by comparing the value of each of the attribute values with all the training data attribute values.And the one having the highest similarity matrix row will make the testing data to fall in the same class. To find the similarity matrix, the algorithm compares the value of each attribute value of the testing data with the same attribute value of the training data and allots '1' if they are identical and '0' if not. This value provides the probability of the testing data tuple to be fallen in each training data tuples.

To store the value of each comparison, the algorithm create m×n matrix, where m is the number of tuple in the training set and n is the number of attribute of testing or training dataset.So the tendency of the testing tuple to fall in will be decided by adding the value of each row of the m×n matrix. The resulting m×n matrix is calledthe *similarity matrix*.The rowhaving the highest value in the similarity matrix will influence the testing tuple to fall in the same class.

But in some cases, different row may have the same highest value in the similarity matrix; in such case the algorithm will select the class label in which majorityof those rows will fall to be the class label of the testing set. But still the number of rows in the same class label may become equal. In such case, k-NN classifier will decide in which class label the testing tuple will fall

Example:
Let us consider the following to be the training data:

Class label

1	0	0	1	0	1	0	0	0	0	1	0	**0**
1	1	1	1	0	0	1	1	0	0	1	0	**1**
1	1	0	0	0	0	0	1	1	1	1	1	**0**
0	0	0	1	1	1	1	0	0	0	1	1	**1**
0	0	0	0	0	0	0	0	1	1	1	1	**1**
1	1	1	0	1	0	1	0	1	0	1	0	**1**

Let us consider the following to be the testing set:

1	0	0	1	0	1	0	1	0	1	0	1

To find the similarity matrix, the algorithm compares each attribute value of testing dataset with each attribute value of training dataset. It puts ‘1’if they are similar and ‘0’ if not.

1	1	1	1	1	1	1	0	1	0	0	0
1	0	0	1	1	0	0	1	1	0	0	0
0	0	1	0	1	0	1	1	0	1	0	1
0	1	1	1	0	1	0	0	1	0	0	1
0	1	1	0	1	0	1	0	0	1	0	1
1	0	0	0	0	0	0	0	0	0	0	0

The algorithm then adds row wise and find the similarity matrix as:

8
5
6
6
6
1

As we can see from the above similarity matrix, first row in the training dataset is very similar in nature to the testing dataset. Therefore the testing dataset will fall in the same class of the first row in the training set which is 0.

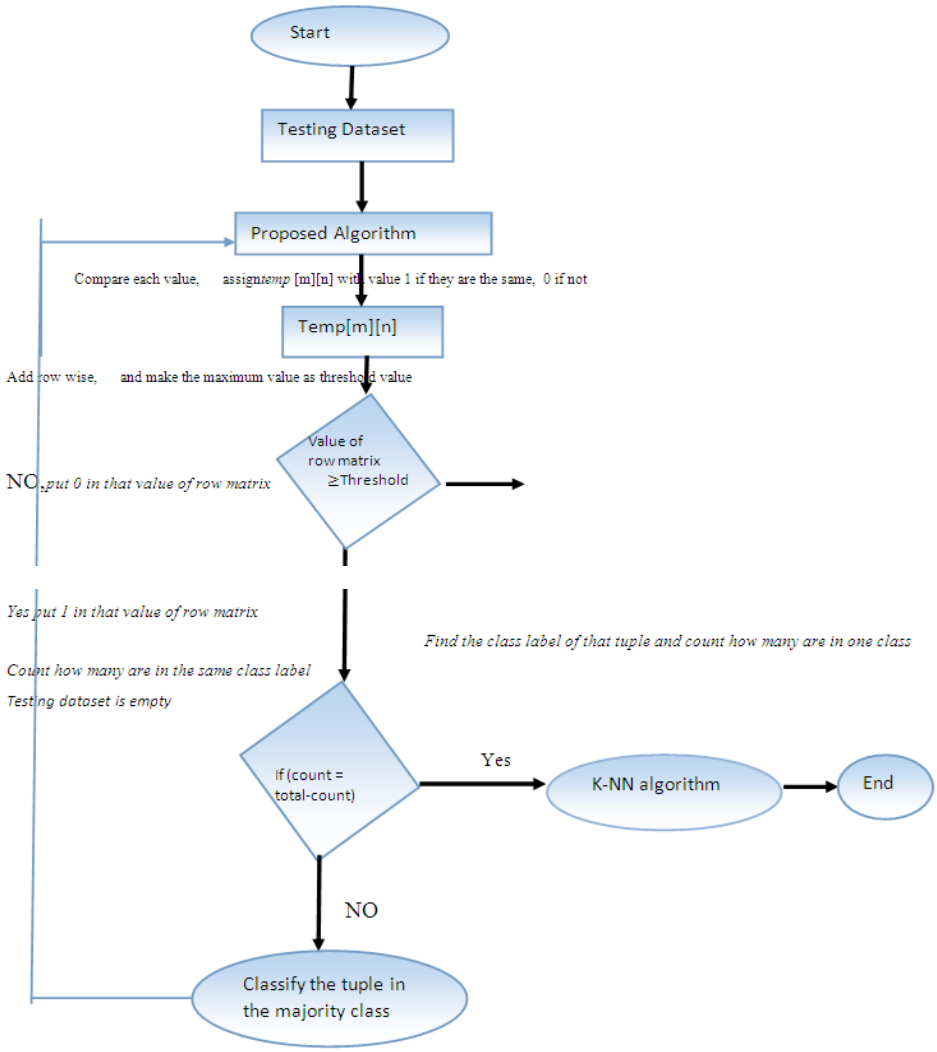

Fig. 3: Flow chart of the proposed system

Algorithm: m× n SimMat classifier

Table 3: Pseudo code of the proposed system

```
1)  Clear all
2)  Load the dataset(both training and test data)
3)  Specify the tuples used as training data from the dataset
4)  select the class label of the training set
5)  Specify the tuple used as test data from the dataset
6)  Select the class label of the test data
7)  Correct_counter=0
8)  Incorrect_counter=0
9)  Specify the size of the test dataset(number of column and row)
10) While (i> number of row in the test dataset)
11) Supper =0
12) Lower=0
13) Make the i^th tuple as tuple to be classified
14) For k=1 to no of column of the test data {
15) For j=1 to the no of row in the training dataset
16) If (the i^th element of the given tuple ==the training element(j,k))
17) Temp1(j,k)=1
18) Else
19) Temp1(j,k)=0
20) End
End
End
21) Sum the result of temp1
22) Transpose it
23) Set the maximum value of the matrix as threshold value
24) For m=1 to the number of element in the transpose matrix
25) If (transpose matrix element at m >=threshold valve)
26) Put 1 in the temp2 at (1,m)
27) Else
28) Put 0 in the temp2 at (1,m)
29) End
30) End
31) Count the number of element
32) For j=1 to the number of element in the temp2
33) If (training tuple class label at (j,1)==1)
34) Increment the number element with value greater than threshold and have
class label of 1
35) Else
36) Increment the number element with value greater than threshold and have
class label of 0
37) End
```

38) If (lower >supper)
39) The algorithm classified the given tuple in class 0
40) Cheek the test label of that tuple is 0 if so
41) Correct_counter=correct_counter+1
42) Else
43) Incorrect_counter=incorrect_counter+1
44) Else if (lower =supper)
45) The data is outlier use K-NN algorithm for some of the tuples which are outlier for this
46) Else
47) The algorithm classifies the tuple in the class 1
48) Check the test tuple class label is 1 is so
49) Correct_counter =correct_counter+1
50) Else
51) Incorrect_counter=incorrect_counter+1
52) End
53) End
54) Display the number of correctly classified tuple \\ disp(Correct_counter)
55) Display the number of incorrectly classified tuple \\ disp(Incorecct_counter)

V. COMPARATIVE ANALYSIS AND PERFORMANCE EVALUATION

For all three algorithms, ten experimentsare conducted by changing the training and testing data. As it is mentioned above, the experiments consider three classification algorithms namely Naive Bayesian, K-Nearest Neighbor and the proposed *m*×*nSimMat*algorithm.These algorithms are compared using accuracy as a measure on the same training and testing datasets which are prepared randomly using making two third of the data as training and the rest as testing.

Before comparing the algorithm efficiency of k-NN with the other, we performed many experiments to find the best value of k at which the algorithm has good efficiency. The following table gives the detail of our work. The k-NN algorithm has the best efficiency on the k values 18, 2, 2, 2, 2, 2, 2, 2, 7, 7 onexperiment number 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 respectively.

Table 4: Efficiency of k-NN with different k values on ten sample datasets.

K	Number of experiment									
	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}	7^{th}	8^{th}	9^{th}	10^{th}
1	63	69	73	76	79	78	78	83	83	85
2	63	69	73	76	79	78	78	83	83	85
3	63	65	67	68	71	69	72	81	87	86
4	60	65	69	70	74	72	72	81	86	87
5	62	61	65	66	68	68	68	77	84	84
6	65	66	70	71	73	74	75	82	88	88
7	66	61	66	68	70	65	70	77	88	88
8	67	63	70	72	71	66	69	76	87	87
9	68	62	65	68	70	66	68	71	84	84
10	67	61	65	68	70	67	70	74	84	85
11	66	61	65	67	68	67	66	73	83	86
12	65	63	66	68	69	68	66	73	83	86
13	65	61	64	68	69	66	65	72	82	85
14	67	62	64	68	69	66	65	72	82	84
15	67	60	63	65	69	63	66	73	81	82
16	66	62	67	70	70	63	68	74	82	84
17	67	59	63	67	66	60	66	72	81	82
18	69	63	62	66	66	62	68	71	81	82
19	67	62	62	65	66	63	68	70	80	82
20	67	63	65	67	68	64	69	71	83	85

From the above table we can conclude that for the given dataset, the k-NN have good efficiency when the value of k is almost 2. The below graph will show at what value of k, the algorithm perform well unambiguously. This graph is directly drawn from the above table using MATLAB.

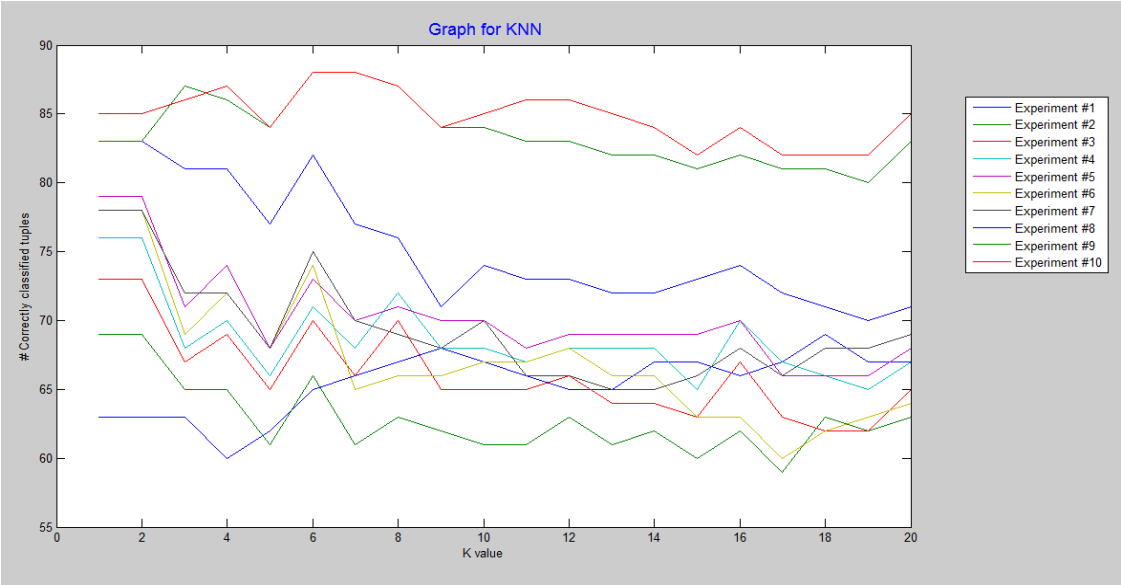

Fig.4: K-NN result for different value of on ten experiment

Therefore for this dataset we prefer the value k be 2 and we take the result of the algorithm when the value of k is 2. Below, we show the result of k-NN, Naive Bayesian and proposed algorithm*m*×*nSimMat* on all ten same sample data from the dataset obtained by randomly grouping.

Table 5: Result of all algorithms on different experiment.

Exp't	**Proposed** *m*×*nSimMat*		**K-NN**		**Naive Bayesian**	
	No. correctly classified tuples	***Accuracy***	Number of correctly classified tuples	***Accuracy***	Number of correctly classified tuples	***Accuracy***
1	65	**73%**	63	**70.8%**	64	**72%**
2	69	**77.5%**	69	**77.5%**	67	**75%**
3	71	**79.8%**	73	**82%**	69	**78%**
4	73	**82%**	76	**85.8%**	69	**78%**
5	79	**88.8%**	79	**88.8%**	69	**78%**
6	79	**88.8%**	78	**87.6%**	66	**74%**
7	78	**87.6%**	78	**88.6%**	67	**75%**
8	83	**93.3%**	83	**93.3%**	70	**79%**
9	84	**94%**	83	**93.3%**	77	**87%**
10	85	**85.6%**	85	**95.5%**	78	**88%**
11	168	**99.4**	144	**85.2%**	144	**85.2%**
Average		**86.35%**		**86.2%**		**79.0%**

From the above table, we can conclude that the newly developed algorithm*m*×*nSimMat* has better performance than both naive Bayesian and k-NN on test on training dataset. Generally for these dataset, the k-NN is more efficient than the rest of the other classification algorithms and the newly developed algorithm*m*×*nSimMat* has even better performance than k-NN. The following graph provides the comparison of the above mentioned classification algorithms.

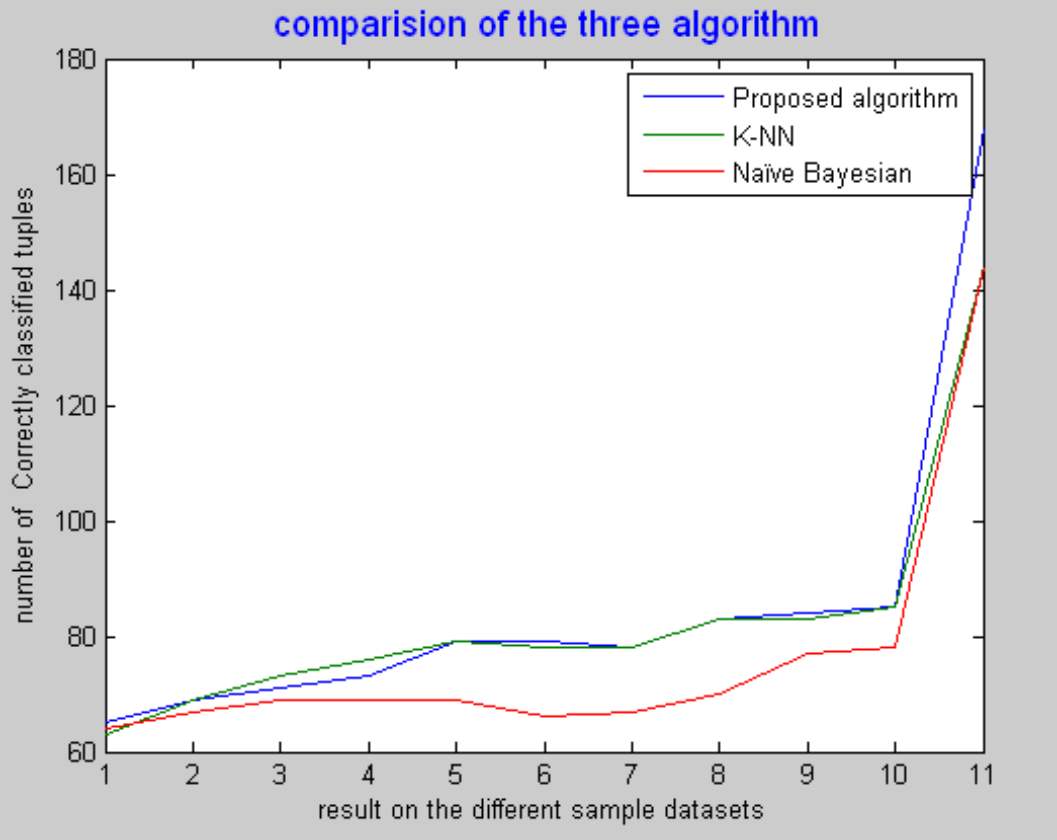

Fig. 5:Comparison of k-NN,Naïve Bayesian and *m*×*nSimMat* algorithms.

VI. CONCLUSION

From the above results,we can conclude that the newly proposed algorithm is efficient in a condition that the test data are not significantly vary from the training dataset. It is sensitive to a noise and outliers but still a more efficient algorithm. The efficiency of one algorithm may outperform the other in one dataset and may have poor efficiency in the other. This implies that without performing as many experiments as required, we cannot say that one classification algorithm is better than the other, except test on training data.

REFERENCE:

[1] Manning C. D. and Schutze H., 1999. Foundations of Statistical NaturalLanguageProcessing [M]. Cambridge: MIT Press.

[2] Yang Y. and Liu X., 1999. A Re-examination of Text Categorization Methods [A].In:Proceedings of 22nd Annual International ACM SIGIR Conference on Research andDevelopment in Information Retrieval [C]. 42-49.

[3] Joachims T., 1998. Text Categorization with Support Vector Machines: Learning with ManyRelevant Features [A]. In: Proceedings of the European Conference on Machine Learning [C].

[4] LiBaoli, Chen Yuzhong, and Yu Shiwen, 2002.A Comparative Study on AutomaticCategorization Methods for Chinese Search Engine [A]. In: Proceedings of the Eighth JointInternational Computer Conference [C]. Hangzhou: Zhejiang University Press, 117-120.

[5] Li Baoli, Yu Shiwen, and Lu Qin. An Improved k-Nearest Neighbor Algorithm for TextCategorization:Department of Computer Science and Technology,Peking University, Beijing,P.R. China, 100871.

[6] R. O. Duda et al, Pattern Classification, 2nd ed. Wiley-Interscience, 2000.

[7] Domingos, P.,and Pazzani, M., (1996), 'Beyond independence: conditions for theoptimality of the simple Bayesian classifier', Proceedings of ICML 1996.

[8] Leung, K., (2007), 'Naive Bayesian Classifier', Technical Report, Department of ComputerScience / Finance and Risk Engineering, Polytechnic University, Brooklyn, New York, USA.

[9] NetworksintheClassificationofTrainingWebPages, IJCSIInternationalJournalof ComputerScienceIssues, Vol. 4, No. 1, 2009.