Evaluating the Performance of Dual Weighted K- Nearest Neighbor Classifier

Pooja Rani
Student
Department of Computer Science and Engineering
GJU S&T, Hisar, India

ABSTRACT
A non-parametric, very simple to use, effective instance-based learning algorithm called K-Nearest Neighbor (KNN), is most widely used to classify the objects in data mining. KNN has some shortcomings which affect its classification performance like the equal impact of all attributes, curse of dimensionality, the value of 'k' parameter and simple voting. A variety of techniques are developed in literature to get better performance. This paper presents an improved algorithm called dual weighted KNN that is a combination of attribute weighted and instance weighted techniques. To verify the performance of proposed algorithm it is conducted on fourteen different datasets taken from UCI in 'R' data mining tool. The results show that the proposed algorithm significantly outperforms than traditional KNN, Attribute weighted KNN and Instance weighted KNN.

Keywords
KNN, Attribute weighted KNN, Instance weighted KNN, and Distance weighted KNN.

1. INTRODUCTION
Data mining is a process of finding useful information from a quantity of data stored in data warehouses or any other data repositories. It uses the classification algorithms to assign the categories of objects from predefined categories. KNN is one of the top ten algorithms used in data mining for classification because it is very simple and very effective algorithm. It is the first choice when we don't have any prior knowledge about the distribution of datasets as it is a non-parametric algorithm. This algorithm was firstly proposed by T.M cover and P.E Hart in 1950 [1] [2]. It is used for classification as well as for regression in various fields like text categorization, ranking models, object recognition, event recognition, medicine, agriculture, finance etc.[3].

KNN is also known as a lazy learner because in its training phase it simply stores all the tuples or do less calculation on the training dataset and awaits until a test tuple is given to it [1] [4].When a test tuple is given, it calculates the similarity between the test tuple and all the training tuples by using distance metrics. the best distance metrics are used like Manhattan distance, Minkowski distance, Mahalanobis distance, Chebyshev distance, and the most commonly used distance metric used in KNN is Euclidean distance metric which gives better results as compared to other distance metrics [5] [6]. The Euclidean distance metric is used when the attributes are not strongly correlated [5] [6]. Let x is a sample whose class label is unknown and y is a set of n training tuples [7] then the Euclidean distance $d(x,y)$ is calculated using the formula :

$$d(x,y) = \sqrt{\sum_{i=1}^{n}\big(a_i(x) - a_i(y)\big)^2}$$

When the distance between test tuple and training tuples is calculated, 'k' number of nearest neighbors are selected and the most common from the neighbors of test tuple 'x' presented by to the test tuple[7]. It is known as class probability estimation method:

$$c(x) = arg\ max \sum_{c\epsilon C\ i=1}^{k} \delta\big(c, c(y_i)\big)$$

Where $y_i = y_1, y_2 \ldots_k$ are the 'k' nearest neighbors of x, 'k' is the number of neighbors and $\delta\big(c, c(y_i)\big)$=1 if c= $c(y_i)$ and $\delta\big(c, c(y_i)\big)$=0 otherwise.

Some key advantages of KNN are: (1) very simple to use, (2) it takes very small time in its training phase, (3) easy to implement and to understand, (4) gives effective results in large datasets as compared to other algorithms, (5) it is independent of data distributions (6) it is robust to noisy data; (7) Effective results when training data is large.

KNN algorithm has some disadvantages as follows: (1) its classification phase takes more time as compared to its training phase; (2) it requires large memory to hoard all training dataset; (3) it is highly dependent on the size of dataset, when the number of samples increases KNN classifier gives the better classification results; (4) it faces a challenging problem to select and decide the appropriate value of 'k' parameter.

The performance of KNN mainly affected by three factors: the value of 'k' parameter, distance metric and the method which is used to predict the class label. Generally, it uses simple majority voting from its nearest neighbors.

KNN is very sensitive to the value of 'k' parameter because the large or small value of 'k' has different characteristics [6]. The selection and deciding the value of 'k' parameter is one of the challenging problems in KNN. When k=1, the simplest form of KNN is called as the nearest neighbor rule [6] [1].

In 1967 Cover and Hart suggest that k=1 is the optimal value for some distribution. If 'k' is very small the classification results are very sensitive to dataset sparseness and noise, vague or mislabeled points, there is a large variation in results which makes it inconsistent to use [8]. If 'k' is very large many outliers within the neighborhood from other class can be selected and it misleads the classification results, also the classifier becomes bias towards the majority class prediction which is another key issue in KNN. Hence, the classification performance of KNN is strongly dependent on the selected value of 'k'.

KNN does not require any prior knowledge about the dataset and it assumes that instances in the datasets are separately and identically distributed, so the instances which are close to each other have the same classification [8]. The simplest majority voting of the class labels for KNN can be a problem if the nearest neighbors vary widely over their distances and the closer ones more reliably indicate the class of the query

object [3]. Computational cost, memory limitations and time-consuming process for searching 'k' neighbors in multimedia of high-dimensional datasets are some another problems in KNN.

The rest sections of this paper are structured as follows: In Section II the related work is briefly discussed and Section III expresses the proposed algorithm Dual weighted KNN. The analysis and results of the experiment are shown in Section IV. The last Section V presents the conclusion and future scope.

2. RELATED WORK

Though KNN is very simple to use and most widely used algorithm, there are number of shortcomings in it. Many researchers and authors have proposed various techniques to overcome these shortcomings. Some of those techniques are described below:

2.1 Distance Weighted KNN

Deciding the value of 'k' parameter is an important part of KNN algorithm but it is a very challenging problem to choose the best value of 'k'. KNN assigns the majority class from its neighbors who create a biasing problem in case when the datasets contain more tuples with major class and fewer tuples with minor class. If the selected value for 'k' is very small then there is a large variation in results and if it is very large the classifier becomes bias towards the majority class. Many researchers suggested many techniques to overcome this problem and purposed some methods which give the best value for 'k.'

One of the methods is a Distance-weighted K-Nearest-Neighbor rule developed by Lan Du *et al* [9]. It uses the dual distance-weighted function. It is robust to different choices of 'k' to some degree and yields good performance with a larger optimal 'k' value.

One another approach is based on instance weighting technique where the instances get more weight on the basis of their closeness to the new instance than such neighbors that are far away from the new instance had been presented by Schliep Hechenbichler [10]. It uses the kernel functions to give the weights to instances based on the distance from the unknown instance. By using this method the effect of 'k' parameter can be reduced up to some extent. Another effective technique is Dynamic K-Nearest Neighbors Naive Bayes with Attribute Weighted method to improve KNN's accuracy was developed by L. Jiang [11]. Eager learning and Lazy learning are combined together to improve the efficiency of the classifier. The best value of 'k' is learned eagerly at training time to fit the training data and at classification time for each given test instance, a local naive Bayes within best k nearest neighbor is lazily built. This technique is very time-consuming to be used in real-world applications.

2.2 Attribute Weighted KNN

The classification of data depends more upon some attributes known as relevant attributes than others. KNN uses Euclidean Distance function to calculate the similarity between test tuple and training tuples. It assumes that all the attributes of an instance have equal importance whether they are useful or not. When many irrelevant attributes are present in a dataset the value of distance becomes inaccurate and influences the classifications results. When many irrelevant attributes are present in a dataset it is known as the curse of dimensionality [1].

We need to reduce the curse of dimensionality to improve the efficiency**,** measurement costs**,** storage costs**,** computation costs of KNN algorithm. It decreases the classification problem and makes it easy for interpretation and modeling. Finding the relevant attributes and making a method in which the relevant attributes get more weight is called attribute weighting. An attribute is relevant to the target concept if changing the value of attribute influence the classification results given by the target concept. If any change in the value of attribute does not influence the classification results then they are called as irrelevant attributes [8]. A weighting method based on information gain and extension relativity was proposed by Baobao *et al* [13]. This method improves the anti-jamming ability and accuracy of the KNN algorithm. The computing time is also reduced. Xiao and Ding suggested a weighting method based on the weighted entropy of attribute value which enhances the accuracy of classification [14]. Li *et al* proposed an attribute weighting method where the irrelevant attributes are reduced and weight is assigned by the method of sensitivity which improves the efficiency of the algorithm [5].

A kind of KNN algorithm based on information entropy attribute weighting method was proposed by Shweta Taneja *et a*l [2]. In this method, while calculating the Euclidean distance between two samples, the information entropy of the sample attribute is examined. This algorithm improves the accuracy of classification however it spent more time in classification when many categorical attributes are given. A new method called Hybrid dynamic k-nearest-neighbor, distance and attribute weighted was devised by Jia Wu *et al* [16]. In this method, the weights are assigned to both attributes as well as instances. It significantly improves the classification performance of KNN.

Attribute weighting is most widely used because of its simplicity, scalability, versatility and good empirical success.

3. PROPOSED ALGORITHM

In the literature review, different variants of KNN have been studied and found that it has many limitations. To remove these limitations an attractive algorithm called dual weighted KNN is proposed. This algorithm takes into account the three main problems in KNN and it enhances the classification accuracy of KNN by reducing these problems:

1) Curse of dimensionality;

2) Sensitivity towards 'k' parameter value

3) Simple majority vote based class prediction

To remove the curse of dimensionality weights are assigned to each attribute based on their importance in classification, by using the method of Information Gain. Information Gain is the difference between information contained in whole dataset and information contained in an attribute. The information gain of attributes is calculated using the concept of entropy[1]. It measures the impurity in a group, higher the entropy more the information content. The information gain $Gain(A)$ is calculated as:

$$Gain(A) = E(T_r) - E(T_r|A) \geq 0$$

Where T_r is a set of training samples, A is the attribute whose Information Gain is to be calculated, $E(T_r)$ is Entropy given as:

$$\mathrm{E(T_r)} = -\sum \mathrm{p_i}\ \log_2\ (\mathrm{p_i})\ ,$$

$$E(T_r|A) = \sum\left(\left|\mathrm{T_{rj}}\right|/|T_r|\right) * \mathrm{E(T_r)}$$

The information gain of each attribute is used as the weight

coefficient. After calculating information gain which is used as the weight coefficient, all the attribute weights are normalized. In the process of Normalization, the lowest attribute weights become zero and they do not take participate in the classification process. Hence, the effect of useless or irrelevant attributes (cures of dimensionality) is reduced. The weights also reduce the equal impact of all attributes as the attributes with large weights have more impact as compared to others having small weights. The Euclidean distance is then changed to weighted Euclidean distance. The weighted Euclidean distance $(d_w(x, y))$ is calculated between test tuple and each training tuple [1]. It can be calculated using the following equation:

$$d_w(x,y) = \sqrt{\sum_{i=1}^{n} w_i\left(A_i(x) - A_i(y)\right)^2}$$

Where w_i (w_1, w_2, w_3 ….,) is the weight of each attribute A_i.

The value of 'k' parameter is then decided manually and the weighted Euclidean distances are sorted in ascending order. To reduce the dependency of KNN on 'k' parameters instances are assigned weights. This will reduce the effect of 'k' parameter value to some degree. To remove the problem of simple majority voting, weights to each nearest neighbor are assigned based on their similarity with the test tuple. The most similar tuple gets more weight as compared to the less similar tuples. The similarity $Sim(x, y)$ between *x* and *y* is calculated using the formula as shown in the equation:

$$Sim(x,y) = 1 - d_w(x,y)$$

Now, to classify the unknown tuple weighted class probability estimation method is used [7].

$$c(x) = \arg\max \sum_{c \in C}^{k} \sum_{i=1} w\,\delta\left(c, c(y_i)\right)$$

Where y_1, y_2, ……y_k are the k nearest neighbors of test instance, k is the number of neighbors, C represents the finite set of class labels and $\delta\left(c, c(y_i)\right)$ = 1 if c =$c(y_i)$ and $\delta\left(c, c(y_i)\right)$ = 0 otherwise**.** Hence, the class label with the combined maximum weight of its votes is assigned as the class label of the unknown tuple.

Flowchart: We have designed a flow chart to make understand and explain our algorithm as shown in Figure.

The proposed algorithm can be explained in following steps as follows:

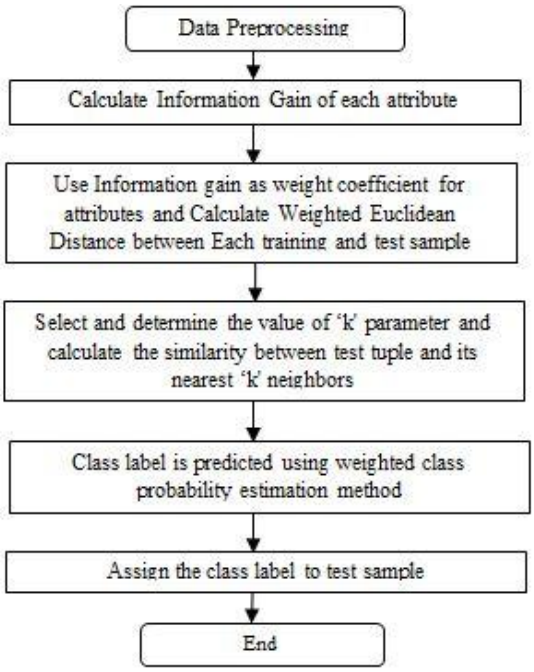

Figure 1 Flow Chart of The proposed System

Step: 1. Read the dataset, which includes a different number of attributes and instances.

Step: 2. Data preprocessing: Normalize the values of each attribute before using the dataset.

Step: 3. Calculating the information gain: The information of each attribute is calculated.

Step: 4. Attribute weighting: Use the information gain as weight coefficients of each attribute.

Step: 5. Normalize the weights.

Step: 6.Distance calculation: Calculated the weighted Euclidean distance between each training tuple and test tuple and arrange the distances into ascending order.

Step: 7. Select the value of 'k' parameter.

Step: 8. Assign the weights for each vote based on the similarity between test tuple and training tuples.

Step: 9. Predict the class label of test tuple using the weighted class probability estimation method.

Step: 10. Assign the class label to the test tuple.

Step: 11. End.

The proposed algorithm performs better than the conventional KNN. Using the attribute weights, the same impact of all attributes and curse of dimensionality is reduced. The instance weighting reduces the effect of outliers and the sensitivity of KNN towards the value of 'k' parameter to some extent. It also improves the simple voting problem as it uses the combined maximum weight to predict the class label of unknown tuple using weighted class probability estimation method. It significantly improves the accuracy of KNN classifier.

4. EXPERIMENTAL ANALYSIS AND RESULTS

In this section, the performance of proposed algorithm is explored in comparison with KNN ((K-Nearest Neighbor), INSWKNN (Instance weighted KNN), AWKNN (Attribute weighted KNN) and proposed algorithm DWKNN in terms of classification accuracy with corresponding 'k' values. The aim of the experiment is to verify the trustworthiness of the proposed algorithm.

4.1 Experiential datasets

The performance of proposed algorithm is tested on fourteen real datasets taken from UCI Machine Learning Repository. The shortened names for following fourteen datasets the Iris, Glass identification, Wine, Image segmentation, Pima Indian diabetes, Ionosphere, Wisconsin Diagnostic Breast Cancer, Sonar, Parkinsons, Herman, Echocardiogram, Banknote, Breast-cancer-Wisconsin and user knowledge modeling are Iris, Glass, Wine, Image, Pima, Iono, Wdbc, Sonar, Parkins, Heber, Echo, Bank, Breast and User respectively. A short description about dataset including the number of instances, number of attributes and number of classes is shown in Table 1.

Table 1 Datasets used in the proposed algorithm

Sr. No.	Dataset	No. of instances	No. of Attribute	No. of classes
1	Iris	150	4	3
2	Glass	214	9	7
3	Wine	178	13	3
4	Image	1500	18	7
5	Pima	541	8	2
6	Iono	351	33	2
7	Wdbc	569	30	2
8	Sonar	208	60	2
9	Parkins	195	22	2
10	Heber	306	4	2
11	Echo	61	11	2
12	Bank	1372	5	2
13	Breast	683	10	2
14	User	258	6	4

4.2 Experimental evaluation method

To evaluate the classification performance of proposed algorithm ten-fold cross-validation method is used. It minimizes the bias of training data. In this method, the datasets are divided into ten equal size parts randomly. One part is used as test dataset at one time and all other 9 parts are used for the training set. The total number of correctly classified tuples in all iterations divided by the total number of tuples in the dataset is calculated as its average accuracy. The classification accuracy analysis is as follows:

The performance of any classifier is evaluated on the basis of the total number of correctly or incorrectly predicted test tuples.

$$Accuracy = \frac{Correctly\ classified\ tuples}{Total\ tuples}$$

4.3 Analysis of the experimental results:

The experiment used different values of 'k' as it is tough to define the best value of 'k'. The results for classification accuracy of KNN, AWKNN, INSWKNN, and DWKNN are provided in Tables 2, 3, 4, 5. In this experiment different 'k' values k=3, k=5, k=7, and k=9 are selected for each dataset. Figures 2,3,4,5 present the classification accuracy of each algorithm and Figure 6 shows the average accuracy of each algorithm.

In Figures 2, 3, 4 and 5 the horizontal axis represents the names of the dataset and the vertical axis represents the classification accuracy of algorithms. Figure 6 shows the average accuracy of each algorithm where the horizontal axis represents the different value of 'k' for each algorithm and vertical axis represent the classification accuracy.

Table 2 Classification accuracy when k=3

Dataset	KNN	INSWKNN	AWKNN	DWKNN
1. Iris	95.33	95.33	95.33	**97.33**
2.Wine	95.49	96.04	96.07	**97.74**
3.Glass	65.93	70.06	68.33	**74.41**
4.Iono	85.2	86.61	85.46	**90.61**
5.Image	94.93	95.33	96.2	**97.01**
6.Pima	70.25	70.79	70.41	**73.18**
7.Wdbc	96.84	96.66	96.66	96.83
8.Sonar	84.11	85.54	85.61	**86.06**
9.Parkins	92.81	92.84	92.21	**94.86**
10.Heberman	70.25	69.25	68.96	**74.19**
11.Echo	90.23	88.33	90.56	**100**
12.Bank	99.78	99.85	99.63	99.63
13.Breast	96.92	96.48	97.07	**97.51**
14.User	82.53	81.4	86.07	**90.72**
Average	87.18571	87.465	87.755	**90.72**

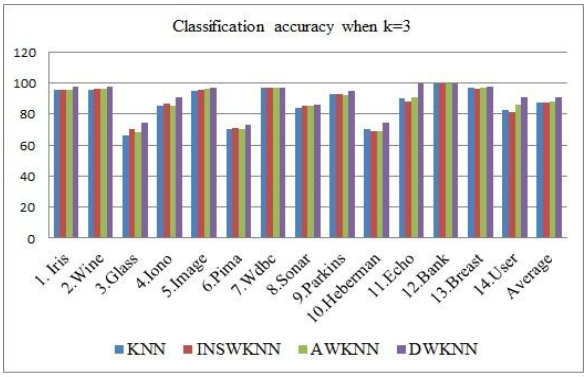

Figure 2 Classification accuracy when k=3

Table 3 Classification accuracy when k=5

Dataset	KNN	INSWKNN	AWKNN	DWKNN
1. Iris	95.33	95.33	95.33	**96.66**
2.Wine	95.49	96.01	96.66	**97.74**
3.Glass	65.92	69.56	67.62	**73.31**
4.Iono	84.87	86.33	84.64	**90.01**
5.Image	94.46	95.66	96.73	**97.57**
6.Pima	69.31	70.59	71.9	**74.63**
7.Wdbc	96.83	96.13	96.84	96.82
8.Sonar	82.64	84.07	82.23	83.66
9.Parkins	92.81	90.73	91.76	**93.68**
10. Heber	69.31	68.21	71.58	**73.87**
11.Echo	93.8	86.66	95.23	**100**
12.Bank	99.85	99.85	99.63	99.7
13.Breast	97.21	96.92	97.21	**98.68**
14.User	81.81	80.26	84.12	**91.46**
Average	87.11	86.87	87.96	**90.55**

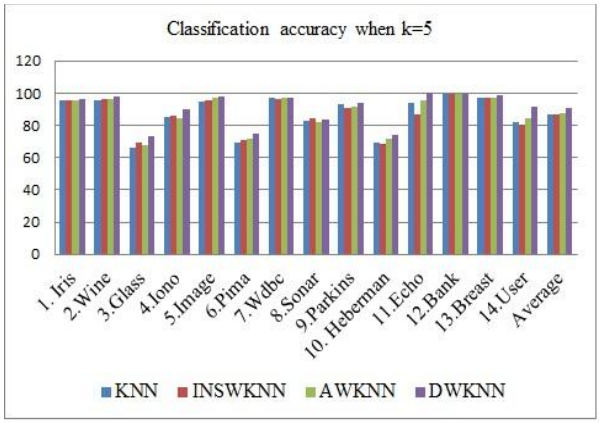

Figure 3 Classification accuracy when k=5

Table 4 Classification accuracy when k=7

Dataset	KNN	INSWKNN	AWKNN	DWKNN
1. Iris	94.66	95.33	95.33	**97.55**
2.Wine	96.63	96.07	96.56	**97.74**
3.Glass	60.84	71.08	67.32	**72.56**
4.Iono	84.03	86.32	84.25	**92.05**
5.Image	94.53	95.6	95	**96**
6.Pima	70.81	70.06	72.83	**76.45**
7.Wdbc	96.65	96.31	96.25	**97.18**
8.Sonar	80.28	85.09	81.73	**87.54**
9.Parkins	92.39	91.28	92.86	**94.31**
10. Heber	69.91	68.62	70.45	**73.56**
11.Echo	93.57	85.47	93.33	**100**
12.Bank	99.85	99.85	99.8	**99.85**
13.Breast	97.36	96.05	97.4	**98.45**
14.User	83.32	80.6	88.4	**91.27**
Average	86.77	86.98	87.96	**91.03**

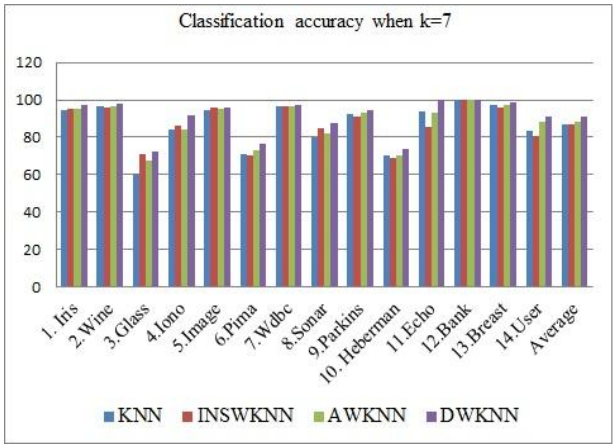

Figure 4 Classification accuracy when k=7

Table 5 Classification accuracy when k=9

Dataset	KNN	NSWKNN	AWKNN	DWKNN
1. Iris	95.33	95.33	95.33	**96.66**
2.Wine	96.07	96.01	96.63	**97.22**
3.Glass	58.74	69.19	62.45	**71.56**
4.Iono	84.31	86.03	83.08	**90.59**
5.Image	94.2	95.06	95.35	**96.75**
6.Pima	71.35	70.8	72.08	**75.22**
7.Wdbc	96.66	96.29	96.49	**97.45**
8.Sonar	76.97	85.09	80.05	**88.96**
9.Parkins	89.15	92.34	89.25	**93.86**
10.Haber	71.86	69.21	71.56	**73.18**
11.Echo	93.33	85.23	93.33	**100**
12.Bank	99.85	99.85	99.72	**99.85**
13.Breast	97.36	96.77	97.11	**98.82**
14.User	80.58	81.41	86.47	**91.84**
Average	86.73	87.04	87.06	**90.85**

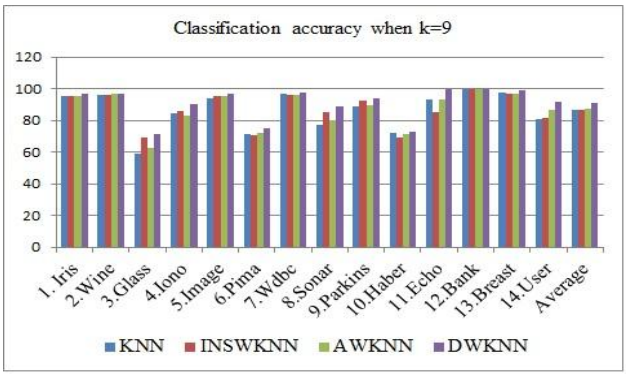

Figure 5 Classification accuracy when k=9

Table 6 Average classification accuracy

Value of 'k'	KNN	INSWKNN	AWKNN	DWKNN
k=3	87.18	87.46	87.75	**90.72**
k=5	87.11	86.87	87.96	**90.55**
k=7	86.77	86.98	87.96	**91.03**
k=9	86.73	87.04	87.06	**90.85**

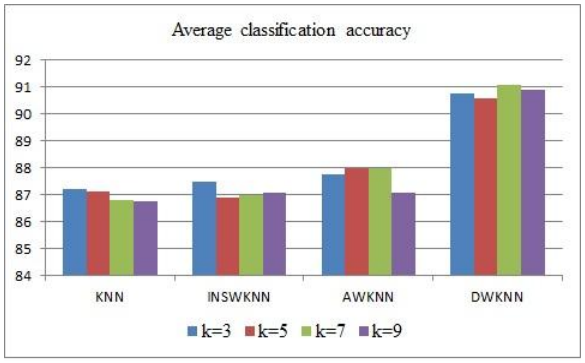

Figure 6 Average classification accuracy

Generally, the value of 'k' should be kept small. If the classes in a dataset are not well separated then a large value of 'k' causes misclassification. The classification accuracy is different for different 'k' values. It is clear from the results that proposed algorithm mostly performs better than KNN, AWKNN, and INSWKNN for each value of 'k'. As the value of 'k' is increased the classification accuracy of the proposed algorithm doesn't go down unlike in traditional KNN. Hence the sensitivity of KNN towards 'k' value is reduced to some extent. The proposed algorithm gives better performance as compared to attribute weighting and instance weighting. It is also clear from results that the proposed algorithm shows the great average increment in classification accuracy when **k=5** and **k=7.** The average accuracy of the proposed algorithm is **90.55** and **91.03** at the value of k=5 and k=7 respectively. The datasets used in the experiment has the different number of attributes and instances. The weighted voting strategy used in an algorithm to predict the class label reduces the biasing towards majority class prediction. The traditional KNN shows the highest average accuracy of **87.77** when k is set to 7. The instance weighted KNN shows **87.46** highest accuracy when k is set to 3.The attribute weighted KNN shows **87.96** highest average accuracies for k=5 and k=7.

5. CONCLUSION

This paper presents an attractive new dual weighted KNN algorithm based on attribute weighting as well as instance weighting technique. The proposed algorithm addresses the problems of KNN such as curse of dimensionality, prediction by simple majority voting, and sensitivity of KNN towards the value selection for 'k' parameter, with the goal of improving the classification performance of KNN. The experiment is conducted on fourteen real datasets taken from UCI. The proposed algorithm not only increases the classification accuracy of KNN it also reduces the sensitivity of KNN towards the selection of 'k' parameter value to some degree. The algorithm performs well for most datasets. The experimental result shows that the proposed algorithm increases the classification accuracy and it has higher average accuracy than the other three KNN variants shown in tables. The proposed algorithm is tested for continues and numeric datasets. For future work the algorithm could be tested for imbalanced and categorical datasets.

6. REFERENCES

[1] J. Han and M. Kamber, Data mining: concepts and techniques, 2nd ed. Amsterdam ; Boston : San Francisco, CA: Elsevier, 2006.

[2] S. Taneja, C. Gupta, K. Goyal, and D. Gureja, "An Enhanced K-Nearest Neighbor Algorithm Using Information Gain and Clustering," presented at the Fourth International Conference on Advanced Computing & Communication Technologies, 2014, pp. 325–329, 2014.

[3] S. B. Imandoust and M. Bolandraftar, "Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background," Int. J. Eng. Res. Appl., vol. 3, no. 5, pp. 605–610, 2013.

[4] D. Wettschereck, D. W. Aha, and T. Mohri, "A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms," in Lazy learning, Springer, 1997, pp. 273–314, 1997.

[5] Z. Li, Z. Chengjin, X. Qingyang, and L. Chunfa, "Weighted-KNN and its application on UCI," presented at the International Conference on Information and Automation, 2015, pp. 1748–1750, 2015.

[6] Ming Zhao and Jingchao Chen, "Improvement and Comparision of Weighted K-Nearest-Neighbors Classifiers for Model Selection," J. Softw. Eng., pp. 1–10, 2016.

[7] L. Jiang, Z. Cai, D. Wang, and S. Jiang, "Survey of improving k-nearest-neighbor for classification," presented at the Fourth International Conference on Fuzzy Systems and Knowledge Discovery, 2007, vol. 1, pp. 679–683, 2007.

[8] M. E. Syed, "Attribute weighting in k-nearest neighbor classification," University of Tampere, 2014.

[9] J. Gou, L. Du, Y. Zhang, T. Xiong, and others, "A new distance-weighted k-nearest neighbor classifier," J Inf Comput Sci, vol. 9, no. 6, pp. 1429–1436, 2012.

[10] K. Hechenbichler and K. Schliep, "Weighted k-nearest-neighbor techniques and ordinal classification," Institue for Statistik sonderforschungsbereich, 386, 2004.

[11] L. Jiang, H. Zhang, and Z. Cai, "Dynamic k-nearest-neighbor naive Bayes with attribute weighted," in International Conference on Fuzzy Systems and Knowledge Discovery, 2006, pp. 365–368, 2006.

[12] D. P. Vivencio, E. R. Hruschka, M. do Carmo Nicoletti, E. B. dos Santos, and S. D. Galvao, "Feature-weighted k-nearest neighbor classifier," presented at the Foundations of Computational Intelligence,2007, pp. 481–486, 2007.

[13] W. Baobao, M. Jinsheng, and S. Minru, "An enhancement of K-Nearest Neighbor algorithm using information gain and extension relativity," presented at the International Conference on Condition Monitoring and Diagnosis, 2008, pp. 1314–1317, 2008.

[14] X. Xiao and H. Ding, "Enhancement of K-nearest neighbor algorithm based on the weighted entropy of attribute value," presented at the Fourth International Conference on Advanced & Communication Technologies, 2012, pp. 1261–1264, 2012.

[15] X. Li and C. Xiang, "Correlation-based K-nearest neighbor algorithm," presented at the 3rd International Conference on Software Engineering and Service Science, 2012, pp. 185–187, 2012.

[16] J. Wu, Z. Hua Cai, and S. Ao, "Hybrid dynamic k-nearest-neighbor and distance and attribute weighted method for classification," Int. J. Comput. Appl. Technol., vol. 43, no. 4, pp. 378–384, 2012.