EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

An Improved Weighted Base Classification for Optimum Weighted Nearest Neighbor Classifiers

Muhammad Abbas[1], Kamran Ali Memon[2,*], Noor ul Ain[3], Ekang Francis Ajebesone[3], Muhammad Usaid[4], Zulfiqar Ali Bhutto[5]

[1]School of Computer Science, Beijing University of Posts and Telecommunications Beijing, China
[2]School of Electronic Engineering, Beijing University of Posts & Telecommunications, Beijing, China
[3]School of Information & Communication Engineering, Beijing University of Posts & Telecommunications, Beijing, China
[4]Department of Electrical Engineering, Mehran University of Engineering & Technology, Jamshoro, Pakistan
[5]Dawood University of Engineering and Technology, Karachi, Pakistan

Abstract

Existing classification studies use two non-parametric classifiers- k-nearest neighbours (kNN) and decision trees, and one parametric classifier-logistic regression, generating high accuracies. Previous research work has compared the results of these classifiers with training patterns of different sizes to study alcohol tests. In this paper, the Improved Version of the kNN (IVkNN) algorithm is presented which overcomes the limitation of the conventional kNN algorithm to classify wine quality. The proposed method typically identifies the same number of nearest neighbours for each test example. Results indicate a higher Overall Accuracy (OA) that oscillates between 67% and 76%. Among the three classifiers, the least sensitive to the training sample size was the kNN and produced the unrivalled OA, followed by sequential decision trees and logistic regression. Based on the sample size, the proposed IVkNN model presented 80% accuracy and 0.375 root mean square error (RMSE).

Keywords: Classification, k-Nearest Neighbor (*k*NN), Logistic Regression, Decision Trees, Cross-Validation, Machine-Learning (ML), SVM, random forest, improved version of k-nearest neighbor (IV*k*NN), and Python.

Received on 22 December 2019, accepted on 26 February 2020, published on 27 February 2020

doi: 10.4108/eai.13-7-2018.163339

*Corresponding author. Email: Ali.kamran77@gmail.com

1. Introduction

This classification is designed to place predefined tags in prior unlabelled examples automatically. It is a predominantly researched area in the fields of information, machine learning and natural language processing. In previous research, several machine learning algorithms have been put forward to process the classification patterns, e.g. *k*NN [1] - [7], SVM [8], [9] and Multinomial Naive Bayes [25].

Traditional *k*NN algorithms are usually premised on the uniform distribution of training patterns in different categories. In many real-world applications, however, inequitable data sets will occur [2]. In a disproportionately weighted data set, most of the classes are represented by the bulk of the examples, while the other few are only a small-scale of all the examples [3]. In the *k*NN-based classification system, *K* is a very critical determinant. Two fundamental classification processes are implored. First, the *k*NN of the test example is determined in the training set; then you establish predictions based on the class labels of the nearest neighbors. General, an asymmetric assignment of examples in each domain of the training set, is established. There may be more examples in some training sets. Consequently, the *K* determinant directly affects the performance of the classification. When the class is large enough, the value of *K* in most cases outcomes a bias [10].

This paper proposes an enhanced version of the algorithm, called IV*k*NN, which ameliorates the performance of conventional *k*NN in hands-on applications, unlike traditional *k*NN algorithms, test

examples are associated with different nearest neighbors, rather than identifying the same number for all examples by the proposed algorithm. IV*k*NN algorithm tested OA on seven data sets. The results of the experimental classification show that IV*k*NN algorithm outmatches the conventional *k*NN. Subsequent sections of this article are structured as follows. Part two presents the conventional *k*NN algorithm. Part three illustrates the IV*k*NN algorithm. Part four presents the experimental results of conventional *k*NN algorithms and the IV*k*NN algorithm. In the end, Part five summarizes this document and provides instructions for future work.

2. The Traditional *k*NN Algorithm

An old and straightforward process in classification is the conventional *k*NN algorithm [4]. Nevertheless, challenging results are often produced, and in some areas, when smartly integrated, improves state of the art significantly [11], [13], [19], [27]. The kNN rule categorizes each of the nearest K neighbors in the training set that are not marked with a majority label. Therefore, its performance mainly depends on the Euclidean metric used to verify the closest neighbors. A simple Euclidean metric is used by most *k*NN classifiers to calculate the differences in the examples given as a vector [14]. The Euclidean distance is obtained by the following formula.

Sometimes we want to calculate the distance between two vectors or points. We have therefore derived some unique removal properties in the Euclidean n-space this way. With some vectors $p, q \in \mathrm{R}^n$, we denote the distance between these two points as follows.
The measure of the connecting line is the Euclidean distance between points p and q given by $\overline{pq}$.

If in cartesian coordinates $p = (p_1, p_2, p_3, \dots, p_n)$ and $q = (q_1, q_2, q_3, \dots, q_n)$ are the two points in Euclidean n-space, then the distance (d) from p to q, or from q to p is given by the Pythagorean formula.

$$d(p,q) = \|p - q\| = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} \quad (1)$$

$$= \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \quad (2)$$

In addition, conventional *k*NN is a highly versatile procedure that is void of pre-processed training data. It provides speed and space benefits for significant problems. Nevertheless, in many hands-on applications, most records do not get good results because the examples are unevenly distributed among the classes. Identifying different numbers of nearest neighbors is appropriate to establish all class labels of the test examples i.e. *k*NN algorithm should be similar. Therefore, the focus of the *k*NN algorithm is to find the best *K* determinant, for each test example to get an appropriate class specification. In this article, we introduce only the traditional *k*NN and IV*k*NN.

3. The Weighted Nearest Neighbor Classifier

The experiment shows an example using a vector space model (VSM) [11], [16]. Where a vector x is represented by each example. The comparability between them is analysed between the two vectors representing each, by the Euclidean distance.

3.1. The 1-nearest neighbor classifier (1nn)

An inherent nearest-neighbor classifier type is a single nearest-neighbor classifier where point x is assigned to the nearest neighbor class in feature space, that is $C_n^{1nn}(x) = Y_{(1)}$, and n is the number.

The most appropriate intuitive classifier that associates point x with the nearest class in feature space, where the most intuitive classifier type for the nearest neighbor is $C_n^{1nn}(x) = Y_{(1)}$, and n is the number.

When the size of the training dataset reaches infinity, the nearest neighbor classifier ensures that the error rate is at least the Bayesian error rate (the least obtainable error rate for a specific data distribution).

3.2. The weighted nearest neighbor classifier

We can think of the k-nearest neighbor classifier as assigning a weight w to the k-nearest neighbor that is $1/k$ greater than all other 0 weights, often generalized to the weighted neighbor classifiers. That is, where the i^{th} nearest neighbor is assigned a weight w_{ni}, with $\sum_{i=1}^{n} w_{ni} = 1$. A steady and equivalent result of the above classifier is also applied.

Let C_n^{wnn} represent the weighted nearest classifier with weights $\{w_{ni}\}_{i=1}^{n}$. The focus on regularity conditions on the distributions of classes of excess risk has the following asymptotic expansion [17]. As we know, the asymptotic expansion is useful when it is truncated to a finite number of terms. The approximation can provide advantages by increasing the speed of the calculation of the extended function. As a rule, the best approximation is given when the series is truncated at the lowest term.

$$\mathcal{R}_{\mathcal{R}}(C_n^{wnn}) - \mathcal{R}_{\mathcal{R}}(C^{Bayes}) = (B_1 s_n^2 + B_2 t_n^2)\{1 + o(1)\} \quad (3)$$

for constants B_1 and B_2 where $s_n^2 = \sum_{i=1}^{n} w_{ni}^2$ and

$$t_n = n^{-2/d} \sum_{i=1}^{n} w_{ni} \{ i^{1+2/d} - (i-1)^{1+2/d} \} \quad (4)$$

The optimal weighting scheme $\{w_{ni}^*\}_{i=1}^n$, that reconciles the two terms with the above performance is determined as follows: set $k^* = \lfloor Bn^{4/d+4} \rfloor$,

$$w_{ni}^* = \frac{1}{k^*} \left[1 + \frac{d}{2} - \frac{d}{2k^{*2/d}} \{ i^{1+2/d} - (i-1)^{1+2/d} \} \right] \quad (5)$$

for $i = 1, 2, \ldots, k^*$ and $w_{ni}^* = 0$ for $i = k^* + 1, \ldots, n$.

At the optimum weight, asymptomatic expansion of additional risk is the dominant term $O\left(n^{-4/d+4}\right)$. The same results are valid when using a wrapped nearest neighbor classifier [12].

3.2. Error rates

There are several results regarding the error rate of the K nearest neighbor classifiers. The k-nearest-neighbor classifier is strong (i.e., consistent for any typical distribution on (X, Y)), assuming $k := k_n$ diverges and k_n/n converges to zero as $n \to \infty$

Let C_n^{knn} denote K nearest neighbor classifier based on a training set of size n. Under certain routine conditions, additional exposure provides the following asymptomatic expansion.

$$\mathcal{R}_{\mathcal{R}}(C_n^{knn}) - \mathcal{R}_{\mathcal{R}}(C^{Bayes})$$

$$= \left\{ B_1 \frac{1}{k} + B_2 \left(\frac{k}{n} \right)^{4/d} \right\} \{ 1 + o(1) \}, \quad (6)$$

for constants B_1 and B_2.

set $k^* = \lfloor Bn^{4/d+4} \rfloor$, provides a compromise between the two terms of the above illustration, for which the nearest neighbor error k^* converges to the Bayes error at the optimal rate (minimax) $O\left(n^{-4/d+4}\right)$.

3.3. Error Rate of the Improved Weighted Base Classification Algorithm

Regarding the error rate of the nearest neighbor K classifier, there are many results. The error rate of the conventional kNN algorithm lies between the Bayesian algorithm and the double Bayesian algorithm. The detailed illustration is as follows in Equation (7).

$$E^* \leq E \leq E^* \left(2 - \frac{C}{C-1} E^* \right) \quad (7)$$

Where E^* is the Bayesian error rate (this may be the lowest error rate), E_{kNN} is represents the error rate of kNN, and equally, c shows the number of classes in all data sets in the above Equation. For $c = 2$, and when the Bayes error ratio E^* turns to zero, this limit is reduced to "below twice the Bayesian error rate". In the nearest K neighbor setup, if the number of data sets goes indefinite, then a higher K values outcome an improved kNN classifier [18].

If $K \to \infty$, the kNN classifier is optimal, and the Bayes error rate is approached infinitely [11]. Therefore, the worst performance is observed very often by the nearest neighbor algorithm.

The preliminary analysis is the error rate of the IVkNN algorithm. Corresponding to the value of the optimal K, for the majority of examples in the training set, they acquire successfully the accurate class labels that identify their optimal K nearest neighbors. Nevertheless, for some examples of learning, for some reason, they cannot get the right grade. As a result, the value of K assigned as optimal K is proportional to the classification accuracy level of the training set. Therefore, the error rate of the training set algorithm kNN approaches the best Bayesian error rate when both the number of training examples x and the number of minimum nearest neighbors K approach infinity, such as x $\to \infty$ and $K \to \infty$ [12].

In the best-case scenario, the example dataset is uniformly and densely distributed over a narrow area. The nearest neighbor is identical for each test case of the training set. In this test case, the class tag is obtained using the same kNN algorithm as the nearest neighbor. This improves the performance of the IVkNN algorithm, and the error rate is indefinitely close to the optimal Bayesian error rate. In the worst case, all test cases use the nearest neighbor algorithm to obtain the class label. This will degrade the IVkNN algorithm and become the nearest neighbor's algorithm. It is not possible to obtain class tags using the same kNN algorithm in all test examples. In practice, the optimum value K for each training example is different. Thus, although the performance of the IVkNN algorithm is improved over the existing kNN algorithm, we can conclude that the error rate is Bayesian and double Bayesian. The error rate of the IVkNN algorithm can be explained in more detail by the following Equation (7).

4. Implementation of the Proposed IV*k*NN

In the implementation of the IVkNN algorithm, the data is first cleaned. Also, a structure is developed to perform an exploratory analysis of the data. Further, a set of functions are used in the proposed model to establish a baseline from which to measure the performance of the proposed model. This article presents the analysis on how to implement and compare with multiple classification models, and make

hyperparameter adjustments to achieve the best model and to analysed the overall test model in the set.

In the histogram diagram, the density level of the actual data gives the quality level of the white wine scale in the interval between (0-10). The quality level of the wine in the histogram seems to be clear, with the actual density of the wine in the total data set being according to the position of the baseline in the histogram in the graph, as shown in Figure 1.

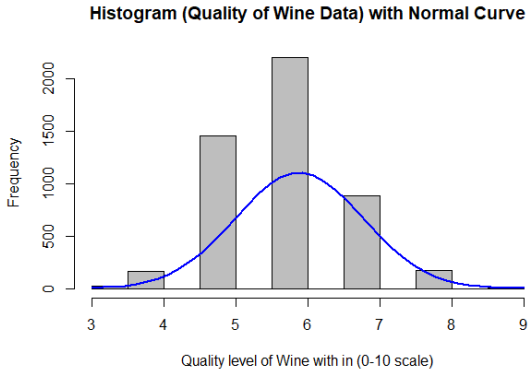

Figure 1. The density of Actual Data

The above in Figure 1. is some summary stats and the distribution plot for wine quality. There are only seven levels of quality out of total ten, and they are following a close normal distribution.

4.1. Split the Wine Quality Data on (train and testing).

In the caret package, the data partition function [24] is used to verify that all datasets contain identical samples for learning the proportions of each class. We normalized all the data and splitted into two classes: training data (eighty-five all data) and test data (fifteen all data), as shown in Figure 2.

To realize the optimal parameters of the classifier, usually, do the verification and training by using the cross-validation. Next, apply a classifier with optimized parameters to the individual test sets to evaluate the effectiveness of the classifier classification. For insufficient data sets, this separation of test data allows a complex trade-off between improving statistical power when evaluating the effectiveness of generalization and choosing the best parameters and choosing the best model.

While the model train, it tries to find some pattern in training data and minimizing the number of errors. Moreover, the proposed model only aims to find a pattern. For the reason that if the model is trying to go beyond the search for patterns, it may memorize the training data.

After training the proposed model, we tested it with recent examples, to avoid problems such as overfitting. We also evaluate how well the proposed model works and faces new data. Depending on performance, we can continue to develop our system [21].

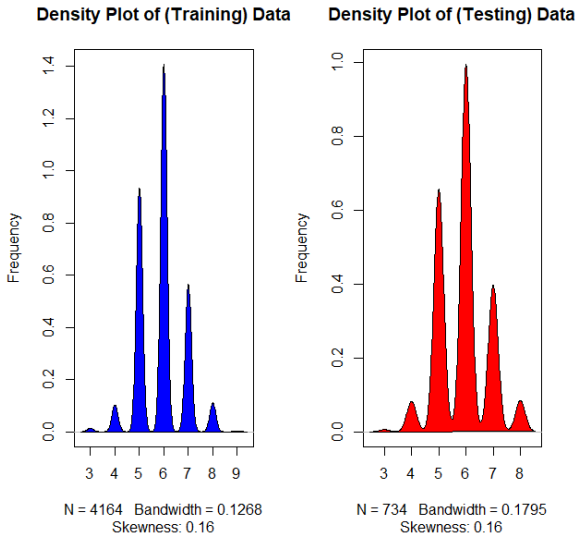

Figure 2. Split the Data into Quality of Wine (Train and Test).

One of the many things required to complete this proposed work is to learn ML use in the scenario. Besides that, is the importance of cross-validation, evaluation of the performance of the model and solution. To solve ML problems, you need to understand cross-validation or resume fully. Cross-validation is used in many ways in machine learning when it is all about parameter and model comparison and selection.

Cross-validation shown in Equ 8. was applied. Cross-validation is an extension of the train/test separation methodology. However, the advantage is that the data set is split randomly several times and trains model tests into slightly different data sets each time. This avoids estimating model errors based on outliers or data that do not accurately represent the signal. Also, it helps prevent over-fitting, over-fitting a model to the characteristics of a data set, and over-fitting on a training set will lead to a terrible fitting on a test set.

Figure 3. depicts k-fold cross-validation, which improves by running it several times. This determines whether the test performance varies depending on the samples to be trained/tested or not. k-fold means choosing the values of *K*, since the selected *K* plays an important role.

Figure 3. *K*-fold cross-validation (10-fold cross-validation)

As the pseudocode of the 10-fold cross-checks, the procedure is used to evaluate each algorithm configured importantly with the same random start number, to ensure that the same divisions are made for the training data and that each of the algorithms is evaluated in the same way [26].

Typically, *K* is selected as 5 or 10. The Lata being very popular in the field of applied machine learning, but there is no conventional rule for it. As *K* spreads out, the size difference between the resampling subset and the training set decreases. As this difference decreases, the method bias reduces as follows:

Steps for K-fold cross-validation

(i) The dataset is divided into *K* equal partitions (or "folds")
(ii) Partition 1 should be used as the testing set and the union of the other partitions as the training set
(iii) Compute the testing precision
(iv) Repeat steps 2 and 3 *K* times, using a different fold as the testing set each time
    a. We will repeat the process 10 times
    b. 2nd iteration
        i. fold 2 would be the testing set
        ii. union of fold 1, 3, 4, 5, 6, 7, 8, 9 and 10 would be the training set
    a. 3rd iteration
        i. fold 3 would be the testing set
        ii. union of fold 1, 2, 4, 5, 6, 7, 8, 9 and 10 would be the training set
    a. Moreover, so on...
(v) Use the mean testing accuracy as the estimate of out-of-sample accuracy

In summary, there are bias-variance trade-offs corresponding to the selection of *K* in the validation process. In general, perform the cross-validate k-folds using $K = 5$ or $K = 10$. Because these values have been shown empirically to obtain an estimate of the test error rate that is not susceptible to an exceedingly high bias or a huge variance.

Quintessentially, the term "cross-validation and cross-testing" enhances this trade-off by reusing test data without displacing the performance of the classifier. And the results showed that this approach has a higher likelihood of tracing meaningful outcomes than the standard cross-validation and testing approach.

If we average the mean square error MSE_{test} for each division, in order to evaluate the model based on several cross-validation divisions, as in Equation (8).

$$CV_{(n)} = \sum MSE_{i,test}/_{n} \qquad (8)$$

4.2. Experimental Results and Comparison with the Existing classifiers

The *k*NN algorithm, being among the most straightforward algorithms used in machine learning to resolve regression and classification problems, employs data and classifiers for new data points based on the identity of measure (for example, the distance function). It should evaluate several different machine learning algorithms on a dataset in Python language, and in these experiments, we compared different regression models, such as *k*NN, logistic regression, decision tree, and Naive Bayes classifiers. As shown in Figures (4, 5, and 6)., different confusion matrices, showing the accuracy estimates for each cross-validation slice for each algorithm separately.

In essence, the key to a fair comparison of machine learning algorithms is to ensure that each algorithm is evaluated equally on the same data, break this data into training and test sets, and then build a model on training data for each algorithm separately based on quality (response variable). Now, as shown in the following Confusion Metric (CM), which is the *k*NN model, and it gave the best result, this is a good point of view of the model's performance, even if it has many weak points, but still its performance is better. When using a confusion matrix after comparing different algorithms and printing the results of our algorithms, the results look like the one shown in Figure 4. This shows that the accuracy of our model is 76.2% of the test dataset.

This study compares the effectiveness of the different classification algorithms for different data sampling strategies and assesses the productiveness of the classification algorithm on the impact and accuracy of the training sample size of the classification [27], [28]. Experimental results are shown in the following graphs, which show the variation in accuracy estimates for each cross-validation model of each algorithm, as we compare in the figures as followed.

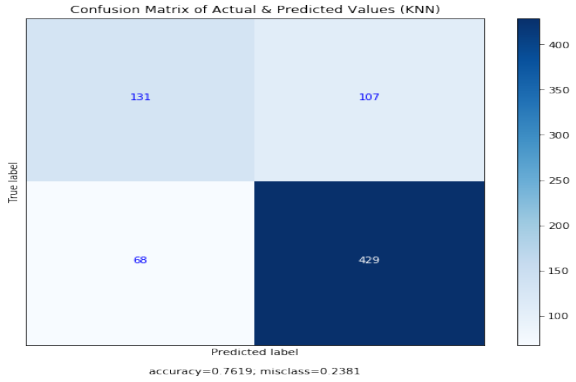

Figure 4. The k-nearest neighbors (*k*NN) CM Result

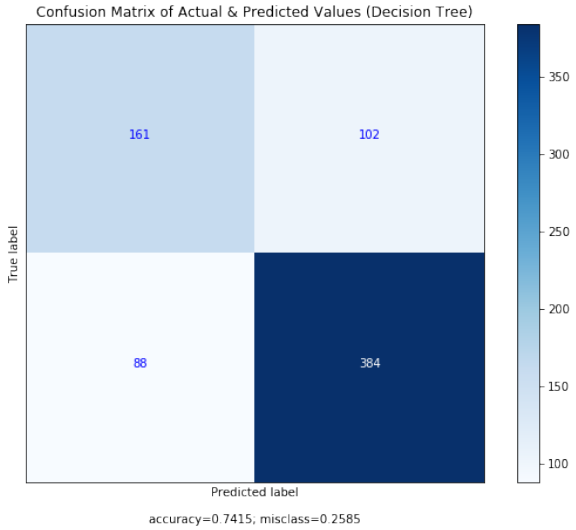

Figure 5. decision tree classifier CM result

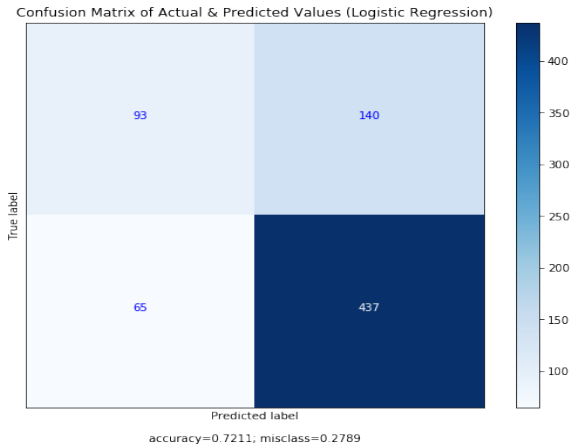

Figure 6. Logistic regression classifier CM result

Based on these results, it can be assumed that both the logistic regression and the linear discriminatory analysis may deserve a further study of this problem. Also, we use a similar set of training (inputs) to compare the functionality of each classifier, verification (verification) data. Because we used the same test data sets for the entire assessment of classification accuracy, to evaluate and compare the performance of classifiers and different data sets because we used OA as a criterion.

As shown in the Table 1., the results between the three algorithms and the compared accuracy of each result of them represent the *k*NN model, which is more accurate than all parameters based on different data sets. The key to a fair comparison of machine learning algorithms is to ensure that each algorithm is evaluated equally on the same data [22]. When performing an additional operation with the *k*NN algorithm using the crosscheck metric for *k*NN, the accuracy is increased from 68% to 76% using the crosscheck metric.

Table 1. The relationship between classifiers (decision trees, logistic regression, and *k*NN) error and accuracy obtained from datasets.

MODELS	Accuracy	RMSE
K-NEAREST NEIGHBOR	0.761905	0.487950
DECISION TREE	0.741496	0.508432
LOGISTIC REGRESSION	0.721088	0.528120

The best precision of the three classifiers was attained when the training sample size was sizable enough since it was determined for large data samples, the algorithms would not work well, but we got a good result, such as logistic regression, decision tree and *k*NN were 72.1%, 74.2% and 76.2%, respectively Table 1. In addition to decision trees this can play a notable role with the use of random forests as random forests construct multiple decision trees and combine them to achieve a more accurate and stable prediction [20]. However, when the size of the training sample increased, the overall accuracy of the classifiers decreased slightly; It is presumed that the size of the training sample is assumed to be large enough to attain the best performance of the classifier if the training sample data is imbalanced between classes. If the size of the training sample is large enough, the ratio between classes may change, decreasing overall precision. In all three classes of this research, the maximum accuracy was achieved when the training sample size was approximately 85% of the whole data in the overall model. It would also have an effect on proposed final model result while reducing the error rate according to Equation 7. Then got the better accuracy which shown in the following Table 2. in the proposed model, and showing the accuracy graph as following in Figure 7.

4.3. Experimental Results of Proposed IV*k*NN Model

As in Table 2., the result shows that the precision of the *k*NN model is excellent compared to others and with the help of the IV*k*NN model get out perform accuracy which is proposed by the *k*NN classifier.

Table 2. Comparison between Existing and proposed Model

MODELS	ACCURACY	RMSE
K-NEAREST NEIGHBOR	0.761905	0.487950
DECISION TREE	0.741496	0.508432
LOGISTIC REGRESSION	0.721088	0.528120
PROPOSED IVKNN	0.80	0.375

As above in table 1., comparison between accuracy and RMSE or RMSD (root mean square deviation). The RMSE value of the predicted values $\hat{Y}_i$ (predicted value) for the times i of the dependent variable of a regression Y_i (actual value) is calculated with variables observed n times for n different predictions as the square root of the mean of the squares of the deviations, as described in Equation (9).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2} \qquad (9)$$

Primarily, the standard error (RMSE) differentiates the predicted value and the known value. It is also known as the standard deviation; lower RMSE values indicate a better fit. As we determined in comparison with the other models, the *k*NN model has a lower RMSE value than others, while decreased the RMSE, the accuracy becomes high with the final proposed model.

The optimal number of neighbors is 34 means 10-fold cross-validation tells us that $K = 34$ results in the lowest validation error.

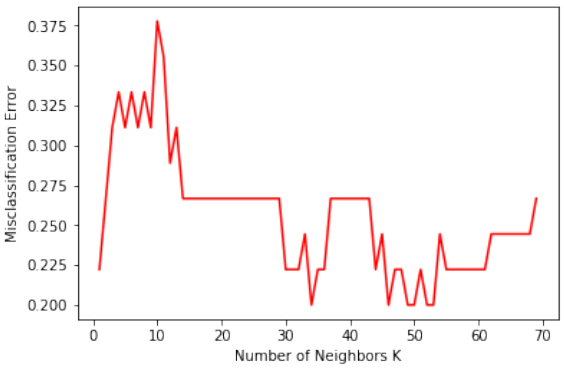

Figure 7. Misclassification Error along with Neighbors *K*

When classifying a single object using a *k*NN classifier, the algorithm establishes K nearest class attributes [23]. Consequently, the value of K plays an essential part in *k*NN performance and is a fundamental parameter tuning algorithm for *k*NN. In this study, a range of K values (1-50 and 70) was identified. to select the optimum classifier parameter *k*NN using different data sets. As we know, *k*NN is not suitable for large data sizes. In such cases, you need to reduce the dimension to improve performance [19]. Also, the processing of missing values that help us in improving the results. As shown in Figure 7. Accuracy increases with increasing K value to 50, and after 50 to 70 accuracy goes to decrease because I analysed that maximum accuracy on $K = 50$, with increasing the K value with high dimension data the problem goes to complex and accuracy does not increase and finally using the proposed IV*k*NN model to get a little good accuracy, and it is tough to get accuracy on extensive data.

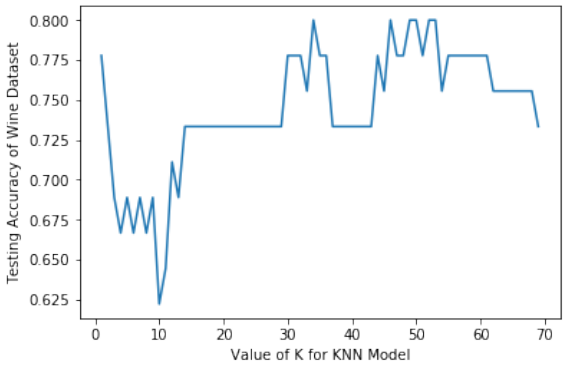

Figure 8. relationship between *K* and testing accuracy

5. Conclusions

This work presents 03 essential contributions. First to overcome the limitation of the traditional *k*NN using over the proposed model. Second, the classification of wine quality was implemented, evaluated and compared with three different machine learning algorithms. Thirdly, this article introduces IV*k*NN, which is a better performance of existing algorithms for reducing the error rate and with the reduction of the error rate an efficient processing power and according to Equation (7). using the optimal weighting scheme as in Equation (5). with the contribution of these improved features the proposed model improved the performance and accuracy to make an efficient processing method for the proposed model, the performance of the proposed model based on these features is better than that of the existential algorithms.

References

[1] Yang, Yiming. "An evaluation of statistical approaches to text categorization." *Information retrieval* 1.1-2 (1999): 69-90.

[2] Japkowicz, Nathalie. "Learning from imbalanced data sets: a comparison of various strategies." *AAAI workshop on learning from imbalanced data sets*. Vol. 68. 2000.

[3] Tan, Songbo. "Neighbor-weighted k-nearest neighbor for unbalanced text corpus." *Expert Systems with Applications* 28.4 (2005): 667-671.

[4] Chien, Y. "Pattern classification and scene analysis." *IEEE Transactions on Automatic Control* 19.4 (1974): 462-463.

[5] Gou, Jianping, et al. "A new distance-weighted k-nearest neighbor classifier." *J. Inf. Comput. Sci* 9.6 (2012): 1429-1436.
[6] Ferreira, João Elias Vidueira, et al. "The use of the k nearest neighbor method to classify the representative elements." *Educación Química* 26.3 (2015): 195-201.
[7] Ferreira, João Elias Vidueira, et al. "Graphical representation of chemical periodicity of main elements through boxplot." *Educación química* 27.3 (2016): 209-216.
[8] Joachims, Thorsten. "Text categorization with support vector machines: Learning with many relevant features." *European conference on machine learning*. Springer, Berlin, Heidelberg, 1998.
[9] Basukala, Amit Kumar, et al. "Towards improved land use mapping of irrigated croplands: Performance assessment of different image classification algorithms and approaches." *European Journal of Remote Sensing* 50.1 (2017): 187-201.
[10] Ma, Hongxing, Xili Wang, and Jianping Gou. "Pseudo nearest centroid neighbor classification." *Frontier Computing*. Springer, Singapore, 2016. 103-115.
[11] Salve, Suhas G., and Kalpana C. Jondhale. "Shape matching and object recognition using shape contexts." *2010 3rd International Conference on Computer Science and Information Technology*. Vol. 9. IEEE, 2010.
[12] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
[13] Simard, Patrice, Yann LeCun, and John S. Denker. "Efficient pattern recognition using a new transformation distance." *Advances in neural information processing systems*. 1993.
[14] Weinberger, Kilian Q., and Lawrence K. Saul. "Distance metric learning for large margin nearest neighbor classification." *Journal of Machine Learning Research* 10.Feb (2009): 207-244.
[15] Mouratidis, Kyriakos, Dimitris Papadias, and Marios Hadjieleftheriou. "Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring." *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005.
[16] Danisman, Taner, and Adil Alpkocak. "Feeler: Emotion classification of text using vector space model." *AISB 2008 Convention Communication, Interaction and Social Intelligence*. Vol. 1. 2008.
[17] Samworth, Richard J. "Optimal weighted nearest neighbour classifiers." *The Annals of Statistics* 40.5 (2012): 2733-2763.
[18] S. Sun, R. Huang, "An Adaptive k -Nearest Neighbor Algorithm," *no. Fskd*, pp. 91–94, 2010.
[19] Khatami, Reza, Giorgos Mountrakis, and Stephen V. Stehman. "A meta-analysis of remote sensing research on supervised pixel-based land-cover image classification processes: General guidelines for practitioners and future research." *Remote Sensing of Environment* 177 (2016): 89-100.
[20] Jhonnerie, Romie, et al. "Random forest classification for mangrove land cover mapping using Landsat 5 TM and ALOS PALSAR imageries." *Procedia Environmental Sciences* 24 (2015): 215-221.
[21] Korjus, Kristjan, Martin N. Hebart, and Raul Vicente. "An efficient data partitioning to improve classification performance while keeping parameters interpretable." *PloS one* 11.8 (2016): e0161788.
[22] Thanh Noi, Phan, and Martin Kappas. "Comparison of random forest, k-nearest neighbor, and support vector machine classifiers for land cover classification using Sentinel-2 imagery." *Sensors* 18.1 (2018): 18.
[23] Qian, Yuguo, et al. "Comparing machine learning classifiers for object-based land cover classification using very high resolution imagery." *Remote Sensing* 7.1 (2015): 153-168.
[24] Kuhn, Max. "Building predictive models in R using the caret package." *Journal of statistical software* 28.5 (2008): 1-26.
[25] Abbas, Muhammad, et al. "Multinomial Naive Bayes Classification Model for Sentiment Analysis." *IJCSNS* 19.3 (2019): 62.
[26] http://karlrosaen.com/ml/learning-log/2016-06-20/
[27] Li, Hu, et al. "Multi-window based ensemble learning for classification of imbalanced streaming data." *World Wide Web* 20.6 (2017): 1507-1525.
[28] Subramani, Sudha, et al. "Deep learning for multi-class identification from domestic violence online posts." *IEEE Access* 7 (2019): 46210-46224.