Innovativity in Modeling and Analytics Journal of Research
vol. 5, 2020, pp.36-49 *iMAJOR*
http://imajor.info/JDA ISSN 2534-9619

Modified k-NN Algorithm with Improved Efficiency

Danail Sandakchiev [1]

Sofia University St. Kl. Ohridski, Sofia, Bulgaria

[1]dsandakchiev@icap.bg

Abstract.
Purpose – The purpose of this paper is to examine a new classification algorithm based on the well-known k nearest neighbors technique that achieves better efficiency in terms of accuracy, precision and time when classifying test observations in comparison to classic k nearest neighbors.
Design/methodology/approach – The proposed methodology splits the input dataset into n folds containing all observations. Each record is allocated to one of the folds. One of the folds is saved for testing purposes and the rest of the folds are used for training. The process is executed n times. The pair of train/test subsets which produces the highest accuracy result is selected as final model for the respective input data.
Findings – 24 different datasets are used for experiments. For each dataset, the classic k-NN is compared to the proposed method (Mk-NN) using accuracy, F1 score and execution time as metrics. The proposed approach achieves better results than classic k-NN according to all used metrics.
Originality/Value – This paper suggests a novel method for improvement in accuracy, precision, recall and time when classifying test observations from a dataset. The approach is based on the concept of k nearest neighbors. However, what separates it from classic k nearest neighbors is that it uses the k-fold method to build a model with the train and test subsets, which best represent the original dataset.

Keywords: Machine Learning Algorithms, Classification problems, k Nearest Neighbors.

1 Introduction

In the realm of machine learning, specifically classification problems, one of the most popular (Lin, Ke and Tsai, 2017) and long-standing techniques is k Nearest Neighbors (k-NN). The method has been around since 1967 (Deng et al., 2018) and since then has gained considerable popularity in both application and research fields. In a nutshell, the technique selects a class for a new observation by finding its k nearest neighbors whose class is known. The nearest neighbors are determined on the basis of their distance (Euclidean, Manhattan or another) to the new data point. Although it is used primarily for solving classification problems, it is not uncommon for k-NN to be applied in pattern recognition, text categorization, object recognition and etc. (Kulkarni, S. and Babu, M., 2013).

The idea of k-NN is applicable to a wide range of problems – business, medicine, media and others. Classic k-NN can be applied in customer relations processes by filtering potential buyers of a specific product or service more effectively, as it can classify them as either buyers or non-buyers (Abdi, Khalili-Damghani and Abolmakarem, 2018). Spatial database is another area where k-NN techniques have an important application. From simply using k-NN to find nearest target points to a query point, to accommodating road networks, obstacles and etc. along with k-NN for more accurate distance evaluations (Taniar and Rahayu, 2013). Social media has become very powerful source of users' data for marketing departments. Knowing as much as possible for your current customers and having effective process for targeting new ones is a key component for the success of many companies. Even though, specific user characteristics (gender, age, marital status and similar) are usually not available publicly, careful selection of appropriate features and inferencing with the help of classifiers, like k-NN, can produce promising predictions for both age and gender, as concluded by a case study on Korean Facebook users (Choi et al., 2017). In the area of software quality, missing data can create serious obstacles when analyzing or modelling a dataset. Imputing the missing data with the help of k-NN has proven to be an adequate alternative solution (Huang et al., 2017). In the field of medicine, an active learning method using fuzzy k-NN has been applied to classify cancer type from microarray gene expression datasets (Halder, Dey and Kumar, 2015). Human emotions are often unpredictable and pose a challenge for machine learning problems. In a study for emotion recognition, six men aged 26-50 were asked to simulate 6 common facial expressions. Muscle movements were recorded and thanks to the data, the scientists managed to build a model for classifying facial expressions using k-NN (Tarnowski et al., 2017).

Many modifications of the classic k-NN algorithm have been proposed since its initial introduction and continue to be proposed to present day, proving its relevance and robustness. Researches are driven most often either by specific case studies or further developments of the algorithm. Weighted k Nearest Neighbor (WkNN) assigns different weights to the k neighbors based on their distance to the test point which makes use of training observations (Dudani, 1976). Similar to WkNN, the Modified k Nearest Neighbor also uses weights, but in addition it also looks for the validity of the data point when classifying nearest neighbor (Parvin et al., 2008). Reduced Nearest Neighbor ignores redundant patterns in the training phase of the classification (Gates, 1972). Model based k nearest neighbor models the input data and uses the model to classify the data. This modification strives to achieve both improved accuracy and more efficient execution time, while also automatically chooses appropriate value for k (Guo et al., 2003). Another modification based on the classic k-NN that tries to improve accuracy without sacrificing cost in time is Rank Nearest Neighbor (kRNN). It suggests assigning ranks to training data for each category (Bagui et al., 2003). A suggested approach for improving accuracy results of classifiers (including k-NN) is creating synthetic attributes which can be utilized in the modelling stage. Synthetic attributes can be created using other different features and data on subjects (Alsaffar, 2017). Image classification accuracy using k-NN based algorithm is proposed to be increased with Nearest Feature Line Neighbor (NFL) which takes advantage of multiple templates per class (Li et al, 2000). Interesting approach, aimed specifically at text recognition, proposes the training sample sets to be clustered with the help of k-means and then use the cluster centers as the actual training samples (Zhou, Li and Xia, 2009).

Although k-NN is very popular machine learning technique for classification problems, it doesn't always produce satisfactory results in terms of efficiency, thus leaves room for improvement in this aspect. Especially, in fields like medicine, accuracy is of crucial importance as poor scores may have severe consequences. A key step in the modelling process is the split of the dataset to train and test. Herein, we propose a new algorithm for applying the concept of k-NN. By searching for the pair of train and test subset which best represents a given dataset, we try to achieve significant improvement in the accuracy results when classifying new observations. Furthermore, the concept of the proposed algorithm implies it would affect positively accuracy, specificity and sensitivity scores regardless of the nature of the input dataset which makes it applicable to wide range of business, humanitarian, social and other areas.

The rest of the paper is organized as follows: in Section 2. we provide a brief review of the classic k-NN. We specify metrics for evaluating the performance of classifiers. We then propose a new modification of k-NN in Section 3. Experiments and Results are presented in Section 4. We provide our conclusions and future work intentions in the final Section 5.

2 Literature Review

One of the most popular and widely used supervised classification techniques is k-nearest neighbors. Often used in a number of areas such as bankruptcy prediction, customer relationship management, fraud detection, intrusion detection, recommender systems and others (Lin, Ke and Tsai, 2017).

When a new test observation is classified, the distance (usually Euclidean or Manhattan) is computed with each training data point. The class of the k nearest train points (neighbors) is considered at next stage. The predominant class within the k nearest neighbors determines the class of the test observation at the end.

The simplicity of k-NN makes it a very attractive tool for solving classification problems at hand. The technique is easily understandable by users with different professional background which also makes it recognizable in a wide range of fields. Results can easily be visualized to an audience. The training process is expected to be very fast and stable even in the presence of noisy data. Furthermore, as with most classifiers, the larger the train dataset, the more effective the classifier would be (Gera and Joshi, 2015; Kulkarni, S. and Babu, M., 2013).

Naturally, k-nn is associated with drawbacks as well. Overall, the computation time is significant and since it is a lazy supervised technique, the testing as well needs considerable time. Using k-NN requires an appropriate amount of space for storage which can turn out to be quite huge. Observations used for training can have big impact on accuracy, so working with representative training dataset is a challenge. Furthermore, the selection of the variables for the model should be done with care, as irrelevant features can easily distort the algorithm (Gera and Joshi, 2015; Kulkarni, S. and Babu, M., 2013). Important parameter, which must be selected when initiating k-NN, is the distance (eg. Manhattan, Euclidean or another) the classifier will use to determine the closest points to a new observation. Applying different distances would yield different accuracy results for a given problem at hand (Ivanov, I. and Tanov, V., 2018)

Probably the biggest challenge when applying k-NN is choosing an optimal value for k. There is still no proven universal method which can select k value for a given problem. Usually, k is optimized by experimenting with different values, as studies show that different datasets would perform best with specific k (Fukunaga, K., and Hummels, D. M., 1987). One rule of thumb applied to this problem is using √n as value for k, where n is the number of observations in the train set. However, experiments show that such rule is more of recommendation for a trial of k then a valid rule (Hassanat et al, 2014). The dependency of the performance of the classifier by the value of k makes the classifier prone to many and different experiments, as well as modifications.

Important aspect of selecting k value which performance best is using metrics and tests to compare how well the method performs with different number of neighbors. Furthermore, as already reviewed there are many modifications of k-NN itself, so metrics and tests are useful also to compare performance among different modifications of k-NN. Moreover, for different datasets, different modifications of k-NN and values of k may give best results. In the below table, we visualize the dimensions of a confusion matrix which will help us define performance measures used for classification problems.

Table 1. Confusion Matrix

	Actual	
Predicted	True Positives (TP)	False Positives (FP)
	False Negatives (FN)	True Negatives (TN)

Positives refer to correctly classified observations, whereas negatives refer to misclassified observations. More specifically, True Positives (TP) gives us the observations which are observed positive and are actually positive, True Negatives (TN) are observed negative and actual negative, False Positives (FP) are observed negatives and actual positives, and finally, False Negatives are observed positives and actual negatives. (Tripathi et al., 2018)

Accuracy as a performance measure looks at the sum of all correctly classified observations divided by the total number of observations in the subset. (Tripathi et al., 2018)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Sensitivity provides information on the accuracy of only positive predictions. (Tripathi et al., 2018)

$$Sensitivity = \frac{TP}{TP + FN}$$

Specificity represents the accuracy only of negative predictions. (Tripathi et al., 2018)

$$Specificity = \frac{TN}{TN + FP}$$

F1 score is a statistical test that takes into consideration both sensitivity and specificity. In other words, it is the weighted average of the sensitivity and specificity. (Tripathi et al., 2018)

$$F1 = 2 * \frac{Sensitivity * Specificity}{Sensitivity + Specificity}$$

Time should not be neglected as a performance factor in solving classification problems. It can make a huge difference to choose the most effective solution for decision making involving enormous databases and complex computations. When using k-NN classifier, it is expected the time needed for training and testing to depend on the number of nearest neighbors used as part of the decision rule. The bigger the value of k is, the more calculations are involved, hence the time needed for executing the method increases.

3 New Modification

Here we propose a new method when building the k-NN classifier. The observations in the dataset are shuffled and then split into n folds containing all observations. Each observation is allocated to one of the folds. One of the folds is used as test data and the rest of the folds as train data. This process runs n times.

For example, let us say we have a dataset with 100 observations. We choose 10 folds. The observations are shuffled and allocated to 10 folds. Every fold has 10 observations. One of the folds is reserved as test data and the observations in the other 9 folds constitute the train data. At the next run the same folds are used, but another fold is selected as test data. The process will take place 10 times and each time a different fold is selected as test data.

Every time a fold is selected as test data we fit k-NN classifier and score the output in terms of accuracy. The accuracy is determined as percent of the correctly classified observations in the test data to the total number of observations in the test data. The observations in the train folds which produce the highest accuracy are saved and used to train the k-NN classifier. Following the fit of the test data, we compute the final accuracy, which should be the same as the highest accuracy during the train process.

We name the described method Mk-NN.

Algorithm: Mk-NN

```
1: Input dataset X;
2: Shuffle data points in X by keeping the initial indexes of each observation;
3: Split X into n folds ($N_{1,}$ $N_2$, $N_3$, $N_4$ … $N_n$);
4: for i=1 to n do:
        4.1. Reserve $N_i$ as Test data points and the rest of the folds use as Train data points;
        4.2. Define value for k in k-Nearest Neighbors;
        4.3. Train and fit k-NN algorithm;
        4.4. Compute accuracy score of the data points in $N_i$ by dividing the number of correctly predicted class of test data points by the total number of test data points;
end for
5: Select the data points from ($N_1$, $N_2$, $N_3$, $N_4$ … $N_n$) producing the highest accuracy score as final test dataset and the rest of the data points in X as train dataset;
6: The train set in step 5 is used in the Mk-NN algorithm;
```

4 Experimental results and discussion

Both the proposed Mk-NN classifier and the classic k-NN classifier are applied to 24 datasets. Both methods are evaluated using accuracy score, F1 score and time. Confusion matrix is used as well to present the result from using both classifiers. With the exception of Marital Satisfaction dataset, all datasets are taken from Knowledge Extraction Evolutionary Learning (KEEL) available at https://sci2s.ugr.es/keel/index.php and UCI Machine Learning Repository available at https://archive.ics.uci.edu/ml/index.php. Marital Satisfaction dataset is available at figshare (https://figshare.com/s/d2bd33a9605a3a204881).

Table 1 gives an overview of the datasets used for the experiments. The selected datasets have a wide range of number of observation, features and classes, and also their expected accuracy score using classic k-NN is below 97%. We consider it impractical to compare the proposed modification Mk-NN with classic k-NN, when there is no real room for improvement in terms of accuracy, as classic k-NN already gives extremely satisfactory results.

Table 1. Datasets Summary

Dataset	Number of observations	Number of features	Number of Classes
Diabetes	768	8	2
German	1000	24	2
Glass	214	9	6
Haberman	306	3	2
Liver	345	6	2
Phoneme	5405	5	2
Vehicle	846	18	4
Marital Satisfaction	7178	11	2
Banana	5300	2	2
Titanic	2201	3	2
Kr-vs-K	28056	6	18
Yeast	1484	8	10

Abalone	4174	8	28
Nursery	12960	8	5
Contraceptive	1473	9	3
Flare	1066	11	6
Wine quality - red	1599	11	6
Wine quality - white	4898	11	7
Ring	7400	20	2
Thyroid	7200	21	3
Chess	3196	36	2
Satimage	6435	36	7
Spambase	4597	57	2
Splice	3190	60	3

4.1 Classifiers

4.1.1.k-NN

For the experiments performed using k-NN, each dataset is split into two subsets (train and test) where the test subset includes 30% of the total number of observations in the respective dataset. The rest of the observations are kept for training. Twelve k-NN classifiers are used with values of k equal to 1, 3, 5, 7, 9, 11, 13, 15, 30, 45, 60 and √n for the classic k-NN. Manhattan distance is implemented when classifying the observations. Random allocation of observations to test and train subsets is used. The procedure is executed ten times for each value of k.

4.1.2.Mk-NN

For the experiments performed using the proposed Mk-NN classifier, each dataset is split into 10 folds where each observation is used and one observation can be allocated to only one fold. Each fold is used once as test subset with the rest of the folds saved for training, hence the classifier runs ten times with unique pair of test/train subsets each time. The observations are shuffled before being distributed to the folds. As with the k-NN experiments, the same twelve values for k are used here, namely 1, 3, 5, 7, 9, 11, 13, 15, 30, 45, 60 and √n. The Manhattan distance is used when classifying the observations. The train/test subset pair with highest accuracy score per k is selected for the final models, where k equals 1, 3, 5, 7, 9, 11, 13, 15, 30, 45, 60 and √n respectively.

4.2.Metrics

4.2.1.Accuracy

For classic k-NN, the accuracy of each classifier (each value of k) is evaluated by computing the average accuracy from all ten runs, whereas the accuracy for Mk-NN is computed using only the train/test pair selected for the final model per each k. The accuracy corresponds to the portion of correctly classified observations from the total observations in the test subset.

4.2.2.F1 score

Apart from accuracy, we also use F1 score to evaluate the results of both classic k-NN and Mk-NN when applied to all datasets. Thus, we are looking how Mk-NN performs in terms of precision and recall compared to k-NN. The score is computed for all 18 datasets and for each value of k (1, 3, 5, 7, 9, 11, 13, 15, 30, 45, 60 and √n).

Since we are using F1 score as available from scikit-learn package in python, *average* is an important parameter for the computation. When there are two classes in the dataset, the parameter is used as it is by default. Otherwise, when we have multiclass targets (more than two classes), we set the parameter to *micro*, which calculates metrics globally by counting the total true positives, false negatives and false positives.

4.2.3.Confusion Matrix

Confusion matrix is presented for six of the datasets, namely these are Nursery, Satimage, Ring, Diabetes, German and Vehicle. The matrixes are compiled using both the classic k-NN and the proposed Mk-NN classifier, where the value for k is 1 in both instances. The confusion matrixes are built on all observations since the number of test observations would differ applying each classifier. Mk-NN uses ten folds to split the dataset, whereas k-NN splits the dataset using 30% percent of it for test cases. Another important note is that since Mk-NN determines one best train/test split used as model for prediction, and k-NN select different subsets on each run. Randomly one result from the ten runs of 1-NN is chosen and confusion matrix is created using it.

4.2.4.Execution Time

For each value of k, the execution time has been recorded using the time library and method available in python 3.6. The experiments were performed on 64-bit Windows 10 with Intel Core i7-7500 and 8.00GB RAM using Spyder 3.2.8 as integrated development environment. It should be noted that the duration for classifying will always vary slightly from one execution to the other, but these differences would not affect the overall conclusions.

4.3.Results

The accuracy results of the experiments are summarized in table 2. It shows the maximum accuracy achieved with classic k-NN and Mk-NN respectively, the number of neighbors with which the maximum accuracy was achieved, and the execution time needed to classify with the k neighbors reaching maximum accuracy. In case, multiple values of k achieve maximum accuracy, we take the execution time from the smallest k value achieving the maximum accuracy. In the two rightmost columns are computed the differences in both accuracy and time between classic k-NN and Mk-NN, again referring to maximum accuracy. The following key points can be noted from the summary table:

- ✓ In all instances, the Mk-NN classifiers outperform the classic k-NN classifiers, except for Heart dataset where the maximum accuracy is the same.
- ✓ There is no specific value for k that predominately gives the best accuracy results.
- ✓ Big improvements in accuracy when comparing classic k-NN and Mk-NN can be observed for Glass, Haberman, Liver and Vehicle, with improvements of 22%, 14%, 11% and 11% respectively.
- ✓ With the exception of Heart dataset, the execution time needed for reaching maximum accuracy is always less using Mk-NN in comparison to classic k-NN
- ✓ Overall, it can be concluded Mk-NN is more efficient than classic k-NN in both accuracy and execution time.

Table 2. MAX Accuracy Results

		k-NN			Mk-NN		Mk-NN vs. k-NN	Mk-NN vs. k-NN
Dataset	**MAX Accuracy**	**k Neighbors**	**Execution Time in seconds**	**MAX Accuracy**	**k Neighbors**	**Execution Time in seconds**	**Accuracy Difference**	**Difference in Time Needed**
Diabetes	0.76	√n	0.09	0.84	5	0.02	**0.08**	-0.07
German	0.73	11, 15, √n	0.20	0.81	45	0.08	**0.08**	-0.12
Glass	0.73	1	0.02	0.95	1, 3	0.00	**0.22**	-0.02
Haberman	0.76	30	0.03	0.90	3, 30, 45, 60	0.00	**0.14**	-0.03
Heart	0.85	11, 15, 60	0.03	0.85	45	0.04	**0.00**	0.01
Liver	0.71	13	0.03	0.82	7	0.02	**0.11**	-0.01
Vehicle	0.67	7	0.08	0.78	7	0.02	**0.11**	-0.06
Phoneme	0.90	1	0.20	0.94	1	0.04	**0.04**	-0.16
Marital Satisfaction	0.80	7, 9 , 15, 30, 45	3.00	0.83	9, 11, 13, 15	0.65	0.03	-2.35
Banana	0.90	9, 11, 13, 15, 30, 45, √n	0.22	0.92	11, 30, 45, 60	0.06	0.02	-0.16
Titanic	0.79	11, 60	0.28	0.85	9, 11, 13, 15	0.08	0.06	-0.20
Kr-vs-K	0.75	7	8.34	0.81	7, 9	1.28	0.06	-7.06
Yeast	0.59	45	0.47	0.67	30, 45, 60	0.09	0.08	-0.38
Abalone	0.27	60	1.36	0.33	30	0.19	0.06	-1.17
Nursery	0.96	7	5.98	0.98	7, 9, 11	0.91	0.02	-5.07
Contraceptive	0.55	30	0.23	0.62	15	0.04	0.07	-0.19
Flare	0.74	9, 13, 30, √n	0.12	0.83	13	0.04	0.09	-0.08
Wine quality - red	0.57	1	0.08	0.67	1	0.01	0.10	-0.07
Wine quality - white	0.57	1	0.23	0.65	1	0.06	0.08	-0.17
Ring	0.71	1	7.53	0.75	1	1.92	0.04	-5.61
Thyroid	0.94	3, 5, 7, 9, 11, 13	6.11	0.95	3, 5, 7, 9, 11, 13	1.59	0.01	-4.52
Chess	0.95	3	2.95	0.98	5, 7	0.61	0.03	-2.34

Satimage	0.91	3, 5, 7	6.91	0.93	3, 5	1.42	0.02	-5.49
Spambase	0.84	1	0.94	0.87	1, 3, 5	0.26	0.03	-0.68
Splice	0.89	60, √n	8.42	0.93	60	1.69	0.04	-6.73

Next, we provide the results from F1 score in table 3 for both classic k-NN and Mk-NN. The table displays the maximum F1 score achieved with classic k-NN and Mk-NN respectively, the number of neighbors with which the maximum F1 score was achieved, and the execution time needed to classify with the k neighbors reaching maximum F1 score. In case, multiple values of k achieve maximum F1 score, we take the execution time from the smallest k value achieving the maximum F1 score. In the two rightmost columns are computed the differences in both F1 score and execution time between classic k-NN and Mk-NN, again referring to maximum F1 score. Looking at the results we can conclude that:

- ✓ For all datasets, Mk-NN always gives the highest F1 score.
- ✓ The best F1 score is always achieved with k=1.
- ✓ The following datasets (8 datasets) have at least 0.10 increase in F1 score when using Mk-NN method compared to F1 score results of classic k-NN: Yeast, Abalone, Contraceptive, Wine quality – red, Wine quality – white, Diabetes and Liver. The biggest improvement can be observed in Abalone dataset – 0.17.
- ✓ In terms of execution time, again Mk-NN provides better results managing to reach higher F1 score in less time in comparison to classic k-NN.

Table 3. MAX F1 Score

		k-NN			**Mk-NN**		**Mk-NN vs. k-NN**	**Mk-NN vs. k-NN**
Dataset	**MAX F1**	**k Neighbors**	**Execution Time in seconds**	**MAX F1**	**k Neighbors**	**Execution Time in seconds**	**F1 Score Difference**	**Difference in Time Needed**
Diabetes	0.87	1	0.03	0.97	1	0.02	**0.10**	-0.01
German	0.94	1	0.08	0.98	1	0.02	**0.04**	-0.06
Glass	0.92	1	0.02	0.99	1	0.00	**0.07**	-0.02
Haberman	0.92	1	0.03	0.98	1	0.03	**0.06**	0.00
Heart	0.94	1	0.03	0.98	1	0.00	**0.04**	-0.03
Liver	0.86	1	0.02	0.97	1	0.00	**0.11**	-0.02
Vehicle	0.90	1	0.05	0.97	1	0.02	**0.07**	-0.03
Phoneme	0.95	1	0.20	0.99	1	0.04	**0.04**	-0.16
Marital Satisfaction	0.94	1	1.14	0.98	1	0.25	0.04	-0.89
Banana	0.96	1	0.11	0.99	1	0.02	0.03	-0.09

Titanic	0.58	3	0.28	0.59	1, 7	0.08	0.01	-0.20
Kr-vs-K	0.86	1	3.38	0.95	1	0.76	0.09	-2.62
Yeast	0.86	1	0.12	0.96	1	0.04	0.10	-0.08
Abalone	0.76	1	0.27	0.92	1	0.13	0.16	-0.14
Nursery	0.97	5, 7, 9	4.86	0.98	1, 5, 7, 9, 11, 13	0.61	0.01	-4.25
Contraceptive	0.81	1	0.06	0.91	1	0.04	0.10	-0.02
Flare	0.78	1	0.08	0.82	1	0.04	0.04	-0.04
Wine quality - red	0.87	1	0.08	0.97	1	0.01	0.10	-0.07
Wine quality - white	0.87	1	0.23	0.97	1	0.06	0.10	-0.17
Ring	0.92	1	7.53	0.98	1	1.92	0.06	-5.61
Thyroid	0.98	1	6.16	0.99	1	1.42	0.01	-4.74
Chess	0.97	1, 3, 5	2.45	0.99	1	0.46	0.02	-1.99
Satimage	0.97	1	3.35	0.99	1	0.86	0.02	-2.49
Spambase	0.94	1	0.94	0.98	1	0.26	0.04	-0.68
Splice	0.92	1	3.95	0.98	1	0.95	0.06	-3.00

Finally, in table 4, table 5, table 6, table 7, table 8 and table 9, we present confusion matrixes built from classic k-NN and Mk-NN for six datasets (Nursery, Satimage, Ring, Diabetes, German and Vehicle) where k has value of 1. The left part of each table displays the confusion matrix as a result of classic k-NN and the right part as a result of Mk-NN. The values on the diagonal starting from the upper left are correctly classified and consequently all other values in a table are incorrectly classified.

Comparing the values which do not belong to the diagonal with correctly classified observation, it is clearly visible that Mk-NN tends to outperform k-NN in terms of accuracy.

Table 4. Nursery Confusion Matrix

1 k-NN	**Actual**			
Predicted	3894	125	31	216
	129	4062	8	123
	41	12	275	0
	125	92	0	3827

1 Mk-NN	**Actual**			
Predicted	4141	24	3	98
	35	4256	1	30
	16	2	310	0
	52	14	0	3978

Table 5. Satimage Confusion Matrix

1 k-NN	Actual					
Predicted	1526	0	6	0	1	0
	0	693	0	3	6	1
	4	1	1301	43	1	8
	1	1	29	564	3	28
	6	1	1	1	682	16
	0	0	5	23	11	1469

1 Mk-NN	Actual					
Predicted	1529	0	3	0	1	0
	0	700	0	0	3	0
	0	0	1351	6	0	1
	1	1	4	611	0	9
	1	1	0	0	703	2
	0	0	3	11	3	1491

Table 6. Ring Confusion Matrix

1 k-NN	Actual	
Predicted	3051	613
	7	3729

1 Mk-NN	Actual	
Predicted	3482	182
	4	3732

Table 7 - Diabetes Confusion Matrix

1 Mk-NN	Actual	
Predicted	489	11
	7	261

1 k-NN	Actual	
Predicted	469	31
	38	230

Table 8 - German Confusion Matrix

1 Mk-NN	Actual	
Predicted	688	12
	10	290

1 k-NN	Actual	
Predicted	655	45
	48	252

Table 9 - Vehicle Confusion Matrix

1 Mk-NN	Actual			
Predicted	215	0	2	1
	0	202	9	1
	2	7	207	1
	0	0	2	197

1 k-NN	Actual			
Predicted	176	29	5	2
	30	181	3	3
	2	6	209	1
	1	4	1	193

5 Conclusion and Future Work

This paper proposes a new method for applying k-NN, which strives to achieve better efficiency in classifying test observations. The method tries to select the most representative train dataset from the respective input dataset by splitting the input dataset in predefined number of folds and checking each fold as test dataset. The pair of train/test datasets that provides the highest score in terms of correctly predicted test observations against all test observations is selected for the final modelling of the dataset in question.

Performing experiments with real datasets shows improved results in terms of accuracy, specificity, sensitivity and execution time compared to classic k-NN. On average, Mk-NN maximum accuracy outperforms classic k-NN by 7% and maximum F1 score has on average 6% improvement when comparing Mk-NN and classic k-NN Whereas the best F1 scores are predominately achieved with 1 nearest neighbor, the best accuracy results don't seem to be related with a specific value of k.

As future work, finding a way to determine the most appropriate number of folds for a given dataset can be explored. Also, experiments with shuffling the input dataset many times and executing the proposed method each time could yield even better accuracy results, albeit increasing execution time.

References

[1] Abdi, F., Khalili-Damghani, K. and Abolmakarem, S. (2018). Solving customer insurance coverage sales plan problem using a multi-stage data mining approach. Kybernetes, 47(1), pp.2-19.

[2] Alsaffar, A. (2017). Empirical study on the effect of using synthetic attributes on classification algorithms. International Journal of Intelligent Computing and Cybernetics, 10(2), pp.111-129.

[3] Bagui, S., Bagui, S., Pal, K. and Pal, N. (2003). Breast cancer detection using rank nearest neighbor classification rules. Pattern Recognition, 36(1), pp.25-34.

[4] Choi, D., Lee, Y., Kim, S. and Kang, P. (2017). Private attribute inference from Facebook's public text metadata: a case study of Korean users. Industrial Management & Data Systems, 117(8), pp.1687-1706.

[5] Deng, Z., Zhu, X., Cheng, D., Zong, M. and Zhang, S. (2018). Efficient k NN classification algorithm for big data. Neurocomputing, 195, pp. 143-148

[6] Dudani, S. (1976). The Distance-Weighted k-Nearest-Neighbor Rule. IEEE Transactions on Systems, Man, and Cybernetics, SMC-6(4), pp.325-327.

[7] Fukunaga, K., and Hummels, D. M. (1987). Bayes error estimation using Parzen and k-NN procedures. IEEE Transactions on Pattern Analysis and Machine Intelligence, (5), 634-643.

[8] Gates, G. (1972). The reduced nearest neighbor rule (Corresp.). IEEE Transactions on Information Theory, 18(3), pp.431-433.

[9] Gera, C. and Joshi, K. (2015). A Survey on Data Mining Techniques in the Medicative Field. International Journal of Computer Applications, 113(13), pp.32-35.

[10] Guo, G., Wang, H., Bell, D., Bi, Y., and Greer, K. (2003, November). KNN model-based approach in classification. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 986-996). Springer, Berlin, Heidelberg.

[11] Halder, A., Dey, S. and Kumar, A. (2015). Active Learning Using Fuzzy k-NN for Cancer Classification from Microarray Gene Expression Data. Lecture Notes in Electrical Engineering, pp.103-113.

[12] Hassanat, A. B., Abbadi, M. A., Altarawneh, G. A., and Alhasanat, A. A. (2014). Solving the problem of the K parameter in the KNN classifier using an ensemble learning approach. International Journal of Computer Science and Information Security, 12(8).

[13] Huang, J., Keung, J., Sarro, F., Li, Y., Yu, Y., Chan, W. and Sun, H. (2017). Cross-validation based K nearest neighbor imputation for software quality datasets: An empirical study. Journal of Systems and Software, 132, pp.226-252.

[14] Ivanov, I. and Tanov, V. (2018). Big Data Analytics Algorithms and Applications. Machine Learnings, ISBN 978-619-239-010-5, Sofia (in Bulgarian).

[15] Kulkarni, S. G. and Babu, M.V. (2013). Introspection of various K-Nearest Neighbor Techniques. UACEE International Journal of Advances in Computer Science and Its Applications, 3, pp.103-6

[16] Li, S. Z., Chan, K. L., and Wang, C. (2000). Performance evaluation of the nearest feature line method in image classification and retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11), 1335-1339.

[17] Lin, W., Ke, S. and Tsai, C. (2017). Top 10 Data Mining Techniques in Business Applications: A Brief Survey. Kybernetes, pp.00-00.

[18] Parvin, H., Alizadeh, H. and Minaei-Bidgoli, B., (2008, October). MKNN: Modified k-nearest neighbor. In Proceedings of the World Congress on Engineering and Computer Science (Vol. 1). Citeseer.

[19] Taniar, D. and Rahayu, W. (2013). A taxonomy for nearest neighbour queries in spatial databases. Journal of Computer and System Sciences, 79(7), pp.1017-1039.

[20] Tarnowski, P., Kołodziej, M., Majkowski, A. and Rak, R. (2017). Emotion recognition using facial expressions. Procedia Computer Science, 108, pp.1175-1184.

[21] Tripathi, D., Edla, D. R., Kuppili, V., Bablani, A., Dharavath, R. (2018). Credit Scoring based on Weighted Voting and Cluster based Feature Selection. Procedia Computer Science 132(2018) 22-31.

[22] Zhou, Y., Li, Y. and Xia, S. (2009). An Improved KNN Text Classification Algorithm Based on Clustering. Journal of Computers, 4(3), pp.230-237.