Entropy Based k Nearest Neighbor Pattern Classification (EbkNN): En-route to Achieving a High Accuracy in Breast Cancer Diagnosis

Pushpam Kumar Sinha[1] and Ankita Sinha[2]

[1] Department of Mechanical Engineering, Netaji Subhas Institute of Technology,
Amhara, Bihta, Patna, India
Email: pushpamsinha1 [AT] gmail.com

[2] Software Engineer 2, Intuit IDC,
Bangalore, India
Email: ankitasinha0811 [AT] gmail.com

ABSTRACT— ***Entropy based k-Nearest Neighbor pattern classification (EbkNN) is a variation of the conventional k-Nearest Neighbor rule of pattern classification, which exclusively optimizes the value of k-neighbors for each test data based on the calculations of entropy. The formula for entropy used in EbkNN is the one that has been defined popularly in information theory for a set of n different types of information (class) attached to a total of m objects (data points) with each object defined by f features. In EbkNN that value of k is chosen for discrimination of given test data for which the entropy is the least non-zero value. Other rules of conventional kNN are retained in EbkNN. It is concluded that EbkNN works best for binary classification. It is computationally prohibitive to use EbkNN for discriminating the data points of the test dataset into number of classes greater than two. The biggest advantage of EbkNN vis-à-vis the conventional kNN is that in one single run of EbkNN algorithm we get optimum classification of test data. But conventional kNN algorithm has to be run separately for each of the selected range of values of k, and then the optimum k to be chosen from amongst them. We also tested our EbkNN method on WDBC (Wisconsin Diagnostic Breast Cancer) dataset. There are 569 instances in this dataset and we made a random choice of first 290 instances as training dataset and the rest 279 instances as test dataset. We got an exceptionally remarkable result with EbkNN method- accuracy close to 100% and better than the ones got by most of the other researchers who worked on WDBC dataset.***

Keywords— Binary classification, Euclidean distance, Training dataset, Information

---

1. INTRODUCTION

Pattern analysis in which class is to be assigned to a given unknown data or test data based on the known classes of training data, and in which no definite form of the mapping function (mapping class to the data) can be found, is most popularly referred to in the mathematical/statistical literature as nonparametric discrimination (NPD). kNN (k-Nearest Neighbor) is the easiest, and yet powerful, machine learning algorithm for NPD [1,2,3]. The seeds of the kNN were sown first in a technical report [4] by Evelyn Fix and J.L. Hodges, Jr. Suppose the classification is between two sets of population: one from data points $a_1, a_2, \ldots \ldots, a_m$ with probability density x and the other from data points $b_1, b_2, \ldots \ldots, b_n$ with probability density y; the assigning of class to a new data point c depends on the likelihood ratio

$$l(c) = x(c)/y(c) \qquad (1)$$

The problem arises with equation (1) when we do not exactly know x, y but instead have to use their estimates. In [4] the authors suggest several estimates of x, y for different problems for both parametric and nonparametric discrimination, like, for example kernel density estimation. In the field of nonparametric discrimination, they were, however, the first in the world to give the nearest neighbor estimates of x, y, and thereby lay the foundation of kNN.
The work done on 1-NN (single nearest neighbor) and kNN in [4] was further extended in the works [5,6,7,8,9,10].
In [10], it was shown that for any sample size n, the single-Nearest Neighbor rule has a lower probability of error than the kNN for certain classes of distributions. [10] also made it clear that there are two extremes of classification problem normally encountered, they are either parametric or nonparametric. The parametric problems are the ones in which the underlying statistical distribution of the observables are known. For the nonparametric problems certain common sense

based approaches may be made with regard to the decision rule, otherwise, strictly speaking there can be no optimal decision rule for this case. [10] states the lemma of the convergence of the nearest neighbor. Let $\{a_1, a_2, \ldots, a_n\}$ be the set of independent identically distributed random variables in the metric space A classified into M classes but with no underlying statistics. Let an unclassified random variable a has as its closest neighbor a_m form the set $\{a_1, a_2, \ldots, a_n\}$. Then a_m tends to a with probability one. This lemma holds for any metric, including the Euclidean, defined over the observables.

There can be several different types of distance metrics that can be computed amongst the data points of the training dataset and that between the query point and the data points of the training data, and the performance of the kNN classification depends on the way the distance metric is defined in particular application [11]. The distance metric most popularly used in the kNN classification is the Euclidean distance which treats the data points as vectors and neglects any statistical relationship that can be fitted to the classification example. However, it has been shown in several works that if the distance metric is trained to learn from the statistical distribution fitting the labelled data, there is an improvement in the performance of the kNN classifier many-folds [12,13,14]. [11] is an advancement over the works by [12,13,14], and in this work the authors propose the learning Mahalanobis distance metric. The learning Mahalanobis distance metric is a linear transformation of the space of the training dataset in such a way that it brings about two key changes in the conventional Euclidean distances calculated between the data points.

1. Firstly the large Euclidean distances between the two particular data points of the same class are optimized in the new metric to show a small distance
2. Secondly the small Euclidean distances between the two particular data points belonging to two different classes are magnified in the new metric to show a large distance

Making the distance metric learn this way ensures that the k nearest neighbors of the unclassified data belong to the same class.

Before we move ahead with studying many more literature to introduce the problem dealt with in this work, let us first summarize the conventional kNN method as it is known today. Consider a typical NPD problem wherein u is the unlabeled data to be assigned the class from the knowledge of training dataset of n data points segregated into m classes, each data point being defined by the value of p features. Then the kNN method is all about finding k nearest neighbors to u from the training dataset for a certain fixed k which can be typically varied from 1 to $\sqrt{n}$ [1]. The neighbors are defined as those data points which are close to each other, the extent of closeness being quantified by the Euclidean distance (Ed) between the data points. The Ed between u and the data point v from the training dataset is calculated by the formula

$$Ed_{u-v} = \sqrt{\sum_{i=1}^{p} (f_{iu} - f_{iv})^2}$$

where f_{iu} is the value of feature i for unlabeled data u and f_{iv} is the value of feature i for the data point v from the training dataset. Such Ed is calculated between u and each of the n data points from the training dataset, denote this set of Euclidean distances $\{Ed_{u-v} | \ v = 1,2, \ldots, n\}$. For a chosen k, the k data points from the training dataset corresponding to the k smallest Euclidean distances from the set $\{Ed_{u-v} | \ v = 1,2, \ldots, n\}$ are grouped together. Then the class of the unlabeled data is the class of the majority of the data points in this group.

Despite its simplicity and power, the conventional kNN method posed two serious questions to the researchers over the years

1. The method is computationally expensive for large sample size and or many-dimensional features because one has to calculate the distances of the unlabeled data from all the data points in the training dataset. Then how does one optimize the computational cost?
2. Which k should one chose for classification? This is obvious that as k is varied in any given particular NPD problem, the performance of the kNN too may vary. It is intuitive that the optimum k which has a higher probability of giving correct classification will vary from one situation to the other, like, for example, even between different unlabeled data points within the same NPD problem.

For achieving computational economy some of the works provided structure to the training dataset [15,16,17,18,19,20,21]. A few of the alternate approaches to achieve computational economy, wherein the focus is on reducing the size of the labeled dataset, are [3,22,23,24,25]. In [22], Hart proposed selecting the subset S from the whole training dataset T such that 1NN (1-Nearest Neighbor) with S discriminates the data points almost as accurately as 1NN does with T. This way he removed the redundant data subset from the whole training dataset, and thereby achieved computational economy. RNN (Reduced Nearest Neighbor rule) [23] is a refinement of CNN (Condensed Nearest Neighbor rule) [22], wherein the data points from the subset S of the whole training data set T selected by CNN are

removed one by one to check whether the resulting reduced subset of subset S discriminates T correctly. [24] proposed selecting three very small training subsets out of the whole training data set, and then classifying the test data in each of these three training subsets by 1NN sub-classifier; the final classification being done by simple voting scheme to achieve a high accuracy. [25] did a careful study that given the training data set T, which data points must be removed from T to get the subset S which discriminates T in a way so as not to affect the generalization accuracy and noise tolerance. They proposed 6 algorithms DROP1-DROP5 and DEL, DROP standing for Decremental Reduction Optimization Procedure and DEL standing for Decremental Encoding Length. DROP1 is derived from RNN [23] with the difference that once a data point is removed from S to get new S it is checked whether this reduced S discriminates itself correctly rather than checking whether this reduced S discriminates T correctly.

Next we are concerned with the problem of optimizing k in kNN. There are two ways to achieve the performance uniqueness in kNN: either get rid of k or fix k for each test or unlabeled data. The work that gets rid of k is by Sinha [26]; he calls the resulting method ANN (Alternative Nearest Neighbor). However, this (getting rid of k) is not the focus of this paper. In this paper we discuss a new algorithm EbkNN (Entropy-based k Nearest Neighbor), which we have found to optimize k for each given test data. But before we do that let us look at literature for some algorithms that optimizes k in kNN. [27] defines a new informative metric that tells how informative a particular data point from the training data set in the vicinity of unlabeled data is. The idea of informativeness is that highly informative points in the vicinity of test point have same class and are far from the points having dissimilar class. Ideally it is required that the informative metric be calculated for all the points in the training data set, but then the assignment of class to the test point based on informativeness will become computationally very expensive. So Song et al in [27] proposed two algorithms of NPD based on informativeness: LI-kNN (Locally informative kNN) and GI-kNN (Globally informative kNN). In LI-kNN firstly k nearest neighbors to the test point are chosen based on Euclidean distances and then out of these k neighbors the class of the most informative point is assigned to the test point. The GI-kNN differs from LI-kNN in that the neighbors are chosen not based on Euclidean distance but based on weighted Euclidean distance in an iterative algorithm. [2] did bootstrap sampling of the training example and combined the technique with nearest neighbor classifier and found that they got better performance than conventional kNN. [28] and [29] conducted text categorization studies with a variety of machine learning algorithms, one among them being kNN. They took k values so large as 30, 45 and 65; and found that the performance of the kNN on Reuters versions 3 and 4 was one of the best. There are a plethora of other literature that solves the problem of optimizing performance of kNN by fixing k [30,31,32,33]. But because our method (EbkNN) is not related in any way to either of these previous works, I merely make a mention of them here and do not discuss them. Also because our work (EbkNN) is neither an advancement over these works [27,2,28,29,30,31,32,33] nor is derived from them, we do not immediately conclude or claim in this paper that EbkNN outperforms a subset of these and is inferior to the rest. Moreover, we do not state in this paper any mathematical theorem giving the lower and upper bounds on the probability of error of EbkNN. We have a strong belief that when NPD problems are to be solved by common sense, any amount of mathematical attempts to justify/rationalize the solution remain inert to the actual performance of the NPD method over real datasets. It is highly likely that what mathematics will infer may not reflect in actual application. So our main focus in this paper is to make the reader understand and appreciate EbkNN method exhaustively. One must take note of the simplicity and power of the method. Thereafter, we also apply the method to real breast cancer dataset and evaluate its performance. Our target is to achieve a high accuracy in breast cancer diagnosis by the EbkNN method. Let us see if we can do that. The breast cancer diagnosis is binary classification between whether the tumor is malignant or benign. Benign tumors are harmless and do not metastasize upon the passage of time, whereas malignant tumors metastasize with the passage of time if left medically unattended to. Ideally we should have combined the features of either of the works [3,22,23,24,25] into our method to take care of computational expenditure, but to keep things simple and to impart a better understanding of the EbkNN method to the reader we have not done so.

2. THE EBKNN METHOD

If there is a set of n different types of information attached to a total of m objects with each object defined by f features/attributes, the entropy s of this set of objects is defined mathematically in information theory [34] as

$$s = -\sum_{i=1}^{n} p_i \log_2 p_i$$

where p_i is the proportion of objects with information i. Interpreting the above information theory for the typical NPD problem in which the m data points are classified into n classes, the classes are the n types of information and the data points are the m objects. Hence the above equation for entropy holds for the stated typical NPD problem with p_i being the proportion of data points of class i.

The EbkNN method is all about selecting a single unique value of k nearest neighbors to classify the query point, and as the name of the method suggests this is done based on the calculations of entropy. Just as we calculated entropy above for the entire training data set, we calculate entropy for each of the individual values of k. So the fundamental idea in EbkNN is that we chose that value of k for classification for which the entropy is the least. However, this fundamental idea needs a little modification, given that for a certain $p_i = 1$ (which means that all the data points grouped together for a certain k has same class), $s = 0$. This means that for k=1, definitely $s = 0$. Therefore, the actual EbkNN method is that we choose that k for classification for which the entropy is the least non-zero value. Hence, 1-Nearest Neighbor decision rule is not integrated into EbkNN. The EbkNN method can be applied to classification of data points into any number of classes. But for the number of classes greater than 2, it is obvious that the method will be computationally very expensive. Hence in this paper we will be applying this method to binary classification only, as convenient simplifications can be done in computation.

In a binary classification problem let, out of k nearest neighbors, q_1 are the number of data points belonging to class 1, and q_2 are the number of data points belonging to class 2. Note that, in this method, the neighbors are determined by calculating Euclidean distances- the conventional distance metric most popularly used in the kNN algorithm. Then for this problem

$$p_1 = {}^{q_1}/_{k}$$
$$p_2 = {}^{q_2}/_{k} = 1 - p_1$$

By varying q_1 and q_2 over a range of natural numbers and doing calculation of entropy it is not hard to observe that as anyone smaller out of the two proportions p_1 and p_2 reduces the entropy reduces. In other words, the greater the deviation between p_1 and p_2, the smaller the entropy. For example consider the following two cases

Case 1: $p_1 = 0.15$, $p_2 = 0.85$
$|p_1 - p_2| = 0.7$
Let entropy for this case be s_1
Case 2: $p_1 = 0.9, p_2 = 0.1$
$|p_1 - p_2| = 0.8$
Let entropy for this case be s_2
It can be inferred immediately that $s_2 < s_1$. This can be verified by calculation.
So, for binary classification problem, one need not explicitly calculate entropy for each value of k. Instead, that value of k be chosen for classification for which the smaller of the two proportions p_1 and p_2 is the least amongst all values of k considered. This is a very big simplification in computation for the EbkNN method. However, this liberty is not available in EbkNN computations for the NPD problem with greater than two number of classes.

For k=m, where m is the sample size of the training dataset classified into two classes, an important question to ask is that ideally what should be q_1 and q_2 or p_1 and p_2 in the EbkNN method. Note that entropy is equal to 1, the maximum, for $p_1 = p_2 = 0.5$. Hence we will choose the training dataset in a particular NPD problem in such a way that m is even and $q_1 = q_2 = m/2$. We do this because then k=m will definitely not be the optimum k for classification.

Before we go ahead and apply the EbkNN method to the actual NPD problem, there is one last issue that needs attention. What is the range of values of k for which we need to calculate entropy or the deviation $|p_1 - p_2|$? k=1 is definitely ruled out. k=2 is also ruled out because then $|p_1 - p_2|$ is either 1 or 0, i.e. entropy is either 0 or 1 respectively. So at the lower end of the range we begin with that k>=3 for which $|p_1 - p_2| \neq 1$. Let us denote this value k_s. For the higher end of the range, suppose that at a particular instance $q_1 > q_2$. Then $p_1 = {}^{q_1}/_{(q_1 + q_2)} > p_2 = {}^{q_2}/_{(q_1 + q_2)}$.
Let the entropy at this instance be E. From this stage as k = $q_1 + q_2$ is increased in increments of 1, further suppose that q_1 remains static where as it is q_2 which increases in steps of 1. As this happens entropy will first increase up to 1 when $q_{2new} = q_1$, q_{2new} being the new value of q_2 after successive increments. Thereafter it (entropy) will reduce, and suppose that it becomes exactly equal to E for $q_{2new} = q_2'$. If p_2' be the new proportion of data points belonging to class 2 for new k = $q_1 + q_2'$, then

$$p_2' = p_1$$
$$\text{or } {}^{q_2'}/_{(q_1 + q_2')} = {}^{q_1}/_{(q_1 + q_2)}$$
$$\text{or } q_2' = {}^{q_1^2}/_{q_2}$$

If $q_1 + q_2' < m$, continue further

It is obvious that the instance k = $q_1 + q_2$ that I am talking about above should be the one such that $|p_1 - p_2|$ is maximum from amongst the subset of set {k| k=1,2,….,m}. For m of the order of hundreds let us take this subset of length 10, i.e., for every set of 10 'k' values beginning from the minimum k = k_s we find that k for which $|p_1 - p_2|$ is maximum and check if $q_1 + q_2' < m$. The moment $q_1 + q_2' \geq m$, that will be the higher end of the range of k for which we need to compare entropy or $|p_1 - p_2|$. Once the optimized k is chosen this way the assigning of class to the unlabeled data is still done based on the majority voting of the classes of data points from amongst this (optimized) group of k nearest neighbor data points.

3. COMPUTER EXPERIMENTS WITH WDBC DATASET

The real dataset on which we run the EbkNN and the kNN method in this work is the WDBC (Wisconsin Diagnostic Breast Cancer) dataset from the UCI machine learning repository. Dr. William H. Wolberg, W. Nick Street, and Olvi L. Mangasarian are the creators of this dataset. There are 569 data points in this dataset, and 32 columns. The first column is the patient ID. The second column is diagnosis: B for Benign and M for malignant. The rest 30 columns have the values of 30 features. However, effectively there are only 10 basic features - radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension. Actually what are these 10 basic features ? During the biopsy procedure to detect whether the abnormal lump or mass of cells formed in the breast is tumor or not, a fine needle aspirate of few numbers of cells from the lump or mass is taken and observed under microscope. During microscopic examination measurements of these 10 basic features of the cell are made. Columns 3 to 12 (in the WDBC dataset) are respectively the mean values of these 10 features, columns 13 to 22 are respectively the standard error (s.e.) of these 10 features and columns 23 to 32 are respectively the worst values of these 10 features. Everyone knows what is meant by mean, so I should not define it here. Standard error (s.e.) is given by

$s.e. = \sigma/\sqrt{n}$

where σ is standard deviation and n is number of sample points in the data. It is actually the difference between accurate mean and the observed mean. Worst is the mean of three of the largest values of concerned feature. Standard error by itself has no meaning; specially it is highly likely that it has directly no role to play in the diagnosis. But with its effect included in mean, it will surely have a role to play in the diagnosis. So now the important question - How do we include the effect of standard error in mean ? We add the mean of a particular feature say feature1_mean to the standard error of that feature say feature1_se to get the upper limit on the value of feature say feature1_ul. Similarly we subtract from the mean of a particular feature say feature1_mean the standard error of that feature say feature1_se to get the lower limit on the value of feature say feature1_ll. Mathematically

feature1_ul = feature1_mean + feature1_se

feature1_ll = feature1_mean – feature1_se

We did so for all the basic 10 features mentioned above, and created a new data file "processed_data.csv". Column 1 of this file is index beginning from 0, column 2 is patient ID, column 3 is diagnosis, columns 4 to 13 is the mean of the 10 features mentioned above, columns 14 to 23 is the standard error of the 10 features mentioned above, columns 24 to 33 is the worst values of the 10 features mentioned above, column 34 is blank, columns 35 to 44 is the upper limit of the 10 features mentioned above, and columns 45 to 54 is the lower limit of the 10 features mentioned above. While calculating the Euclidean distances, both for EbkNN and kNN, we considered only the features in columns 24 to 33, columns 35 to 44, and columns 45 to 54.

While selecting the training subset out of a set of 569 instances, we could have used the results of either of the works [3,22,23,24,25] . But we did something else. Because 569 is a big number we considered that the training subset so large as a little over 50% of the whole set should be sufficient to capture almost all possible variations of values of features. Then we will run EbkNN and kNN over this subset as training data and the rest as test data. In next step, whatever misclassifications we get in the first step will be removed from the test data and included in the training data. This sounds somewhat like [22], but is far different from it. The major difference being that the training subset is chosen randomly in this work, given the large number of instances.

First 290 instances (145 benign and 145 malignant instances) are chosen as training data and the rest 279 instances are test data. We call it experiment 1. What performance measure is the most important in diagnosis of disease ? It is accuracy and the accuracy of any diagnostic tool should ideally be 100%. Hence, in this work too the only performance measure we will be interested in is accuracy. And our aim in this work will be to take accuracy as close as possible to 100%.

4. RESULTS

4.1 Experiment 1

The accuracy of EbkNN method = 92.1147 %. The optimized value of k for EbkNN method which we compute for assigning class to the test data point by majority voting varied from so small as 5 to as large as 98, the results of all 279 test data points considered.
The accuracy of kNN (k=5) method = 92.1147 %

4.2 Experiment 2

The misclassified data points in EbkNN for experiment 1 are data point nos. 298, 299, 341, 348, 364, 376, 380, 386, 407, 414, 431, 449, 466, 473, 477, 482, 492, 509, 514, 533, 537, and 542.
The misclassified data points in kNN for experiment 1 are data point nos. 298, 299, 341, 348, 364, 376, 380, 386, 407, 422, 431, 466, 473, 477, 482, 492, 509, 514, 519, 533, 537, and 542.
We now include these misclassifications in training data and remove them from the test dataset. So, for both EbkNN and kNN, the training data has 312 data points and the test data has 257 data points. Note that in EbkNN method for experiment 2 the number of benign samples (=162) and the number of malignant samples (=150) are not equal. So, a natural question to ask is that if this skew can cause the optimum k to be 312 for any one particular test data. Let us convince ourselves that this will never be the case here before we go ahead. The biggest $|p_1 - p_2|$ (for meaning of p_1 and p_2, refer section 2) can be as small as 0.34 if optimum k = 3 in the application of EbkNN method to any dataset, which is definitely greater than $|p_1 - p_2|$ value (=12/312=0.03846) for optimum k = 312 in the application of EbkNN method to this dataset here in experiment 2.
The accuracy of EbkNN method = 98.8327 % (3 errors from amongst 257 test data points)
The accuracy of kNN (k=5) method = 98.0545 % (5 errors from amongst 257 test data points)

4.3 Experiment 3

If there are large number of features defining the data points of a given dataset, we can combine these features in a manner so that each combined component (called the principal component) is orthogonal to every other combined component. This process of feature extraction or dimensionality reduction is called Principal Component Analysis (PCA) [1]. We will not go into the details of PCA as this is not the focus of this paper. But the reason that I am talking about it here is that we also did PCA of the data file "processed_data.csv" with entries in columns 24 to 33, 35 to 44, and 45 to 54; and the number of principal components into which the features were extracted were 5, 10 and 15 respectively. We thereafter chose first 290 instances of "processed_data.csv" as training data and the rest 279 instances as test data and classified the test data with the application of the EbkNN method. Then we also included the misclassified test data in training dataset (as in experiment 2) and ran the EbkNN method again. The accuracies that we obtained for all the three PCA components 5, 10 and 15, were far less than that obtained in experiment 2. Hence we do not make a mention of the results of this experiment here.

5. CONCLUSIONS

The accuracies of the EbkNN method and kNN (k=5) method are same for experiment 1, where as in experiment 2 the accuracy of the EbkNN method was better than that of the kNN (k=5). This means that in going from experiment 1 to experiment 2 there was no over-fitting of the data in EbkNN; EbkNN showed better noise tolerance than kNN. It is possible that for both experiment 1 and experiment 2, the kNN method may perform the best for k other than 5. But for this we have to run the kNN algorithm separately for a range of values of k. This is the disadvantage of the conventional kNN algorithm, which we have overcome in our newly found EbkNN method. In single run of the EbkNN method we pick the optimum k separately and exclusively for each test data. Anyhow we already found a high enough accuracy, as close to 100% (98.8327%) as possible, with the EbkNN method. This accuracy is the best from amongst the works of other researchers on WDBC dataset [35,36,37,38,39,40,41,42,43] that the authors of this paper know of. The work by [43] is the closest that it gets to our work. Their best accuracy is 98.62% for kNN with chi-square based feature selection. We are sad that we failed to further improve our accuracy of 98.8327% with PCA. PCA seldom fails and it is amazing that PCA failed here.

It is not so that we did not do feature selection/extraction in experiment 1 and experiment 2. Calculating the upper limit values and the lower limit values of the 10 basic effective features in WDBC dataset was a part of the process of feature extraction only. No other researcher till now has done so with the WDBC dataset and we proudly claim that we are the first one in world to do so, and it reaped fruits for us. The feature extraction that we did in experiment 1 and

experiment 2, however, was not the one which reduced number of features. Why the EbkNN method gave such promising result with the WDBC dataset is a matter for investigation by mathematicians and statisticians alike.

It was serendipity that we discovered high accuracy of the EbkNN method in breast cancer diagnosis. However, it does not follow that the EbkNN method will demonstrate similar performance for other datasets. In fact, the performance of any machine learning algorithm over any dataset is dependent on the internal statistical structure of the dataset. Hence for different datasets the machine learning algorithm which shows the best performance will be different.

6. REFERENCES

[1] S. Dutt, S. Chandramouli and A.K. Das. Machine Learning. Pearson India Education Services Pvt Ltd. Noida. 2020.
[2] Y. Hamamoto, S. Uchimura, and S. Tomita, "A Bootstrap Technique for Nearest Neighbor Classifier Design," IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, vol. 19, no. 1, pp. 73-79, 1997.
[3] E. Alpaydin, "Voting Over Multiple Condensed Nearest Neoghbors," Artificial Intelligence Review, vol. 11, pp. 115-132, 1997.
[4] E. Fix and J. Hodges. Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties. USAF School of Aviation Medicine, Report No 4, 1951.
[5] M. V. Johns, ‘ (An empirical Bayes approach to non-parametric two-way classification,” in Studies in Item Analysis and Predic- tion, H. Solomon, Ed. Stanford, Calif.: Stanford University Press, 1961.
[6] L. N. Kanal, “Statistical methods for pattern classification,” Philco Rept., 1963; originally appeared in T. Harley et al,, “Semi-automatic imagery screening research study and expen - mental investigation,,” Philco Reports, VO43-2 and VO43-3, Vol. I, sec. 6, and Appendix H, prepared for U. S. Army Electronics Research and Development, Lab. under Contract DA-36-039-SC- 90742. March 29. 1963.
[7] G. Sebestyen, Decision Making Processes in Pattern Recognition. New York: Macmillan, 1962, pp. 90-92.
[8] Nils Nilsson, Learning Machines. New York: McGraw-Hill, 1965, pp. 120-121. D. 0.
[9] Loftsgaarden and C. P. Quesenberry, “A nonparametric estimate of a multivariate density function,” Annals Math Stat., vol. 36, pp. 1049-1051, June 1965.
[10] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," IEEE Trans. Inform. Theory, vol. IT-13, pp. 2127, 1967.
[11] K. Q. Weinberger and L. K. Saul, "Distance Metric Learning for Large Margin Nearest Neighbor Classification," Journal of Machine Learning Research, vol. 10, pp. 207-244, 2009.
[12] S. Chopra, R. Hadsell, and Y. LeCun,” Learning a similiarty metric discriminatively, with application to face verification,”.In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-05), pages 349–356, San Diego, CA, 2005
[13] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov,”Neighbourhood components analysis,” In L. K. Saul, Y. Weiss, and L. Bottou, editors, Advances in Neural Information Processing Systems 17, pages 513–520, Cambridge, MA, 2005. MIT Press.
[14] S. Shalev-Shwartz, Y. Singer, and A. Y. Ng,” Online and batch learning of pseudo-metrics,” In Proceedings of the Twenty First International Conference on Machine Learning (ICML-04), pages 94–101, Banff, Canada, 2004.
[15] J.K. Uhlmann ,” Satisfying general proximity/similarity queries with metric trees,” Information Processing Letters; 40: 175-179, 1991.
[16] T. Liu, A.W. Moore and A. Gray,” New Algorithms for Efficient High-Dimensional Nonparametric Classification,” Journal of Machine Learning Research; 7: 1135-1158, 2006.
[17] R.F. Sproull,” Refinements to Nearest Neighbor Searching in *k*-Dimensional Trees,” Algorithmica; 6:579-589, 1991.
[18] S.Z. Li, K.L. Chan and C. Wang,” Performance Evaluation of the NFL Method in Image Classification and Retrieval,” IEEE Trans on Pattern Analysis and Machine Intelligence; Vol 22-Issue 11,2000.
[19] Y. Zhou and C. Zhang ,”Tunable Nearest Neighbor Class,” Pattern Recognition ;346-349 pp, 2007.
[20] Y.C. Liaw, C.M. Wu and M.L. Leou,” Fast Exact k Nearest Neighbors Search using an Orthogonal Search Tree,” Pattern Recognition; Vol 43-Issue 6: 2351-2358, 2010.
[21] J. McNames,” Fast Nearest Neighbor Algorithm based on Principal Axis Search Tree,” IEEE Trans on Pattern Analysis and Machine Intelligence;Vol 23-Issue 9:964-976, 2001.
[22] P. Hart, "The Condensed Nearest Neighbour Rule," IEEE Transactions on Information Theory, vol. 14, pp. 515-516, 1968.
[23] G. Gates, "The Reduced Nearest Neighbour Rule," IEEE Transactions on Information Theory, vol. 18, pp. 431-433, 1972.

[24] M. Kubat and M. Jr, "Voting Nearest-Neighbour Subclassifiers," in Proceedings of the 17th International Conference on Machine Learning, ICML-2000, Stanford, CA, pp. 503-510.,2000.
[25] D. R. Wilson and T. R. Martinez, "Reduction Techniques for Instance-Based Learning Algorithms," Machine learning, vol. 38, no. 3, pp. 257-286, 2000.
[26] P. K. Sinha, "Modifying one of the machine learning algorithms kNN to make it independent of the parameter k by re-defining neighbor," I.J. Mathematical Sciences and Computing, 4, 12-25, 2020.
[27] Y. Song, J. Huang, D. Zhou, H. Zha, and C. L. Giles, "Iknn: Informative k-nearest neighbor pattern classification," in Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases, Berlin, pp. 248-264., 2007
[28] Y. Yang, "An evaluation of statistical approaches to text categorization," Information Retrieval, vol. 1, pp. 69-90, 1999.
[29] Y. Yang and X. Liu, "A re-examination of text categorization methods," in Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval, Berkeley, pp. 42-49, 1999.
[30] M. Jirina and M. J. Jirina, "Classifier Based on Inverted Indexes of Neighbors," Institute of Computer Science, Technical Report No. V-1034, 2008.
[31] M. Jirina and M. J. Jirina, "Using Singularity Exponent in Distance Based Classifier," in Proceedings of the 10th International Conference on Intelligent Systems Design and Applications (ISDA 2010), Cairo, pp. 220-224, 2010.
[32] M. Jirina and M. J. Jirina, "Classifiers Based on Inverted Distances," in New Fundamental Technologies in Data Mining, K. Funatsu, Ed. InTech, vol. 1, ch. 19, pp. 369-387, 2011.
[33] A. B. Hassanat, M. A. Abbadi, and G. A. Altarawneh, "Solving the problem of the K parameter in KNN classifier using an Ensemble Learning Approach," International Journal of Computer Science and Information Security, Vol. 12, No. 8, pp. 33-39, 2014.
[34] C.E. Shannon, "A mathematical theory of communication," Bell System Technical Journal, 27(3): 379-423, 1948.
[35] S. Srivastava, N. Sharma, S.K. Singh, and R. Srivastava," Quantitative analysis of a general framework of a CAD tool for breast cancer detection from mammograms," J Med Imaging Health Inform, 4, 654-74, 2014.
[36] S. Saini, and R. Vijay," Mammogram analysis using feedforward back propagation and cascade-forward back propagation artificial neural network," In 2015 Fifth International Conference on Communication Systems and Network Technologies IEEE, pp 1177-80, 2015.
[37] M.M. Pawar, and S.N. Talbar," Genetic fuzzy system (GFS) based wavelet co-occurrence feature selection in mammogram classification for breast cancer diagnosis," Perspect Sci, 8, 247-50, 2016.
[38] S.J.S. Gardezi, I. Faye, F. Adjed, N. Kamel, and M.M. Eltoukhy," Mammogram classification using curvelet GLCM texture features and GIST features," In International Conference on Advanced Intelligent Systems and Informatics Springer Cham, pp 705-13, 2016.
[39] K. Vaidehi, and T.S. Subashini,"Automatic characterization of benign and malignant masses in mammography," Procedia Comput Sci, 46, 1762-9, 2015
[40] J. Harefa , A. Alexander, and M. Pratiwi," Comparison classifier: support vector machine (SVM) and K-nearest neighbor (K-NN) in digital mammogram images," Jurnal Informatika dan Sistem Informasi, 2, 35-40, 2017.
[41] M. Pratiwi, J. Harefa , and S. Nanda,"Mammograms classification using gray-level co-occurrence matrix and radial basis function neural network," Procedia Comput Sci, 59, 83-91, 2015.
[42] H. Rajaguru, S. Chakravarty S.R., "Analysis of Decision tree and k-nearest Neighbor algorithm in the classification of breast cancer," Asian Pac J Cancer Prev, 20 (12), 3777-3781, 2019.
[43] Z. Mushtaq, A. Yaqub, S. Sani & Adnan Khalid," Effective K-nearest neighbor classifications for Wisconsin breast cancer data sets," Journal of the Chinese Institute of Engineers, pp 1-13, DOI: 10.1080/02533839.2019.1676658, 2019.