LDBNN: A Local Density-based Nearest Neighbor Classifier

Joel Luís Carbonera[a]
Institute of informatics, Federal University of Rio Grande do Sul, Porto Alegre, Brazil

Keywords: Machine Learning, Classification, Instance-based Learning.

Abstract: K-Nearest Neighbor (KNN) is a very simple and powerful classification algorithm. In this paper, we propose a new KNN-based classifier, called local density-based nearest neighbors (LDBNN). It considers that a target instance should be classified in a class whose the k nearest neighbors constitute a dense region, where the neighbors are near to each other and also near to the target instance. The performance of the proposed algorithm was compared with the performance of 5 important KNN-based classifiers. The performance was evaluated in terms of accuracy in 16 well-known datasets. The experimental results show that the proposed algorithm achieves the highest accuracy in most of the datasets.

1 INTRODUCTION

The KNN classifier is one of the most attractive nonparametric techniques in pattern classification because of its simplicity, intuitiveness and effectiveness (Wu et al., 2008; Gou et al., 2014). It has been applied in a wide range of fields, for example, for text classification (Yong et al., 2009), for estimating forest parameters from remote sensing data (Reese et al., 2005), for predicting economic events (Imandoust and Bolandraftar, 2013), for different kinds of diagnosis in medicine (Sarkar and Leong, 2000; Deekshatulu et al., 2013), for various tasks in astronomy (Ramírez et al., 2001; Li et al., 2008), etc.

Although KNN-based classifiers often achieve good classification performance in many practical applications, there are some well-known issues that affect its performance. One of the main issues regarding the approach is its sensitiveness to the choice of the neighborhood size k. The performance of the algorithm can produce very different results, depending on the choice of k and the structure of the data (including the presence of noise). The second issue is how to combine the information about the neighbors of a target instance to support its classification. Originally, the KNN algorithm considers that the target instance is classified according to the class of the majority of its neighbors. Regarding this point, it is important to notice that a given instance can be more or less representative of its class, and this can affect its capability of supporting the classification of a novel instance.

In order to deal with the issues previously mentioned, a number of variations of the KNN-based approaches have been developed (Dudani, 1976; Mitani and Hamamoto, 2006; Gou et al., 2011; Gou et al., 2014). In this paper, we propose a new KNN-based classifier called *local density-based nearest neighbors* (LDBNN). This approach assumes that the target instance i should be classified in the class c whose nearest neighbors to i occupy a dense region around i. That is, the neighbors within class c should be closer to i and closer to each other. This strategy makes the LDBNN algorithm more noise-resistant, since, in general, noisy instances do not constitute dense regions (they tend to be sparsely distributed) (Ester et al., 1996).

In order to evaluate the LDBNN algorithm, we compared its accuracy in a classification task with other 5 algorithms, in 16 well-known datasets. The results show that LDBNN algorithm provides the highest average accuracy in the considered scenarios. Also, it achieves the highest accuracy in most of the datasets. This suggests that the *local density* is a powerful concept that can be further investigated for developing better classification algorithms.

Section 2 presents some related works. Section 3 presents the notation that will be used throughout the paper. Section 4 presents our approach. Section 5 discusses our experimental evaluation. Finally, Section 6 presents our main conclusions and final remarks.

[a] https://orcid.org/0000-0002-4499-3601

Carbonera, J.
LDBNN: A Local Density-based Nearest Neighbor Classifier.
DOI: 10.5220/0010402003950401
In *Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS 2021) - Volume 1*, pages 395-401
ISBN: 978-989-758-509-8

2 RELATED WORKS

In this section, we will discuss some important classification algorithms that are based on the ideas underlying the KNN algorithm.

According to (Murty and Devi, 2011), the NNC (*nearest neighbor classifier*) (Cover and Hart, 1967) constitute one of the simplest classification procedures. It basically classifies the target instance according to the class of its *nearest neighbor*. One of the main drawbacks of this strategy is its high sensitivity to noise. The presence of noise has negative impacts on its accuracy (Aggarwal and Reddy, 2014).

The KNN (*k-nearest neighbors*) (Altman, 1992; Wu and Ai, 2008) can be viewed as a generalization of the NNC. This algorithm classifies a target instance i in the class that includes the majority of the k nearest neighbors of i. When $k = 1$, the KNN is equivalent to NNC. Since it considers more information about the neighborhood of the target instance, the KNN is less sensitive to noise than NNC. However, noisy instances produce negative impacts also in the performance of KNN. Besides that, the performance of KNN is highly sensitive to the choice of the value of k; the number of neighbors that it should consider.

The LMKNN (*local mean-based k-nearest neighbor*) (Mitani and Hamamoto, 2006), adopts the notion of *local centroid* for avoiding the harmful effects of noise in the classification. A local centroid is basically an abstraction (an average) extracted from a set of neighbors of the target instance i. The first step of the algorithm is to identify the set NN of the k nearest neighbors of i. After, the algorithm extracts one local centroid for each set of neighbors that have the same class. Finally, LMKNN classifies the target instance i in the class of its nearest local centroid. The adoption of local centroids makes LMKNN more noise-resistant than KNN. Due to this, in general, the accuracy of LMKNN is higher than the accuracy achieved by KNN.

The WKNN (*distance-weighted k-nearest-neighbor*) (Dudani, 1976) adopts a weighted voting strategy for classifying the target instance i. In this strategy, instead of simply counting the class with more neighbors within the k nearest neighbors of i, the algorithm determines a weight for each class c, which is calculated as the sum of the weight of each neighbor of i in c. At the end, the algorithm classifies i in the class with the lowest weight. The weight w_j of a given neighbor n_j is a function of its distance to i, such that

$$w_i = \begin{cases} \frac{d_k^{NN} - d_j^{NN}}{d_k^{NN} - d_1^{NN}}, & d_k^{NN} \neq d_1^{NN} \\ 1, & d_k^{NN} = d_1^{NN} \end{cases} \tag{1}$$

, where j represents the ascending order of the neighbor n_j (which is the j-th nearest neighbor of i); and d_j^{NN} represents the distance between the target instance i and the j-th nearest neighbor of i. Thus, notice that the closer a neighbor is from the target instance i, the greater is its weight. The goal of this strategy is reducing the effects of the parameter k in the performance of the algorithm.

The DWKNN (*dual weighted voting k-nearest neighbor*) (Gou et al., 2011), can be viewed as an improvement of WKNN. It also adopts a weighted voting strategy for classifying the target instance i. The DWKNN also classifies i in the class with the greatest weight, which is determined as the sum of the weights of the individual neighbors of i that belongs to that class. The weight w_j of a given neighbor n_j is a function of its distance to i, such that

$$w_i = \begin{cases} \frac{d_k^{NN} - d_j^{NN}}{d_k^{NN} - d_1^{NN}} \times \frac{1}{j}, & d_k^{NN} \neq d_1^{NN} \\ 1, & d_k^{NN} = d_1^{NN} \end{cases} \tag{2}$$

Notice that the main difference between WKNN and DWKNN is regarding the inclusion of an additional term in the formula that defines the weight of a neighbor. This term increases the weights of the nearest neighbors of i. The results show that this subtle change significantly improves the accuracy of the algorithm, compared with the WKNN.

The LMPNN (*local mean-based k-nearest neighbor*) (Gou et al., 2014) also adopts the notion of *local centroid* used by LMKNN (Mitani and Hamamoto, 2006). At a first step, the LMPNN identifies the k nearest neighbors of i in each of the p classes used to classify the dataset. Thus, the set of nearest neighbors has $k \times p$ neighbors, instead of just the k neighbors considered by other algorithms. After, for each class, the algorithm generates k subsets of the k nearest neighbors identified, in a way that each subset includes the j nearest neighbors of i, where $1 \leq j \leq k$. After, the algorithm extracts one local centroid from each one of the k subsets, in a way that the j-th local centroid abstracts the information about the j nearest neighbors of i. Thus, the algorithm produces k local centroids for each class. After, the LMPNN calculates the weight w_j of each local centroid lc_j (j-th local centroid) of each class, in a way that

$$w_j = d(i, lc_j) \times \frac{1}{j} \tag{3}$$

That is, the weight of a given local centroid is a function of its distance to the target instance i and its order j (in an ascending ordering of distance to the target instance). The goal of the second term of the formula is to reduce the influence of farthest local centroids. Finally, the algorithm also calculates the weight of each class as the sum of the weights of its local centroids. With this information, the LMPNN algorithm classifies the target instance i in the class with the lowest weight. This algorithm is highly noise-resistant and, in general, it provides a higher accuracy, in comparison with the other algorithms discussed in this section.

In this paper, we propose an algorithm that also decreases the effects of noise by adopting the notion of *local density*.

3 NOTATIONS

In this section, we introduce the following notation that will be used throughout the paper:

- $T = \{x_1, x_2, ..., x_n\}$ is a non-empty set of n instances (or data objects). It represents the training dataset.
- $U = \{x_1, x_2, ...\}$ is a non-empty set that includes every possible instance. Thus, this set is potentially infinite. Notice that $T \subseteq U$.
- Each $x_i \in U$ is a $m-tuple$, such that $x_i = (x_{i1}, x_{i2}, ..., x_{im})$, where x_{ij} represents the value of the j-th feature of the instance x_i, for $1 \leq j \leq m$.
- $L = \{l_1, l_2, ..., l_p\}$ is the set of p class labels that are used for classifying the instances in T, where each $l_i \in L$ represents a given class label.
- $c\colon L \to 2^T$ is a function that maps a given class label $l_j \in L$ to a given set C, such that $C \subseteq T$, which represents the set of instances in T whose class is l_j. Notice that $T = \bigcup_{l \in L} c(l)$. In this notation, 2^T represents the *powerset* of T, that is, the set of all subsets of T, including the empty set and T itself.
- $d\colon U \times U \to \mathbb{R}$ is a distance function that maps two instances in a real number that measures the distance (or dissimilarity) between them. This function can be domain-dependent.

4 LDBNN ALGORITHM

As previously mentioned, the LDBNN (*local density-based nearest neighbors*) algorithm is based on the notion of *local density*. Intuitively, it assumes that a given target instance i should be classified in the class whose k nearest neighbors constitute a *dense* region around i. In this context, a dense region is a region where the instances are near to each other. It is *local* in the sense that the density is being evaluated in the context of the region that includes the k nearest neighbors of i of each class.

Figure 1 represents the notion of local density adopted in this work. It represents a target instance i and the 3 nearest neighbors of the 3 classes of the training set: A, B and C. Notice that the training set can have much more instances, but in this context, our analysis is considering just these 9 instances, and the density of the instances and classes are considered only in this context. In this example, the region occupied by the class A is locally denser than the regions occupied by the classes B and C. Besides that, this density is distributed around the target instance i, since all the neighbors of the class A are close to i. In this situation, our approach assumes that the instance i should be classified in class A.

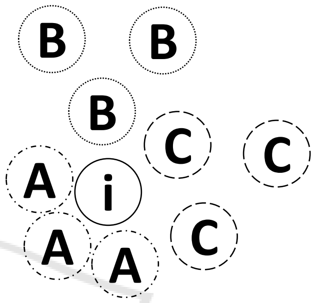

Figure 1: Illustration of the notion of local density. The region occupied by the class A is locally denser than the regions occupied by the classes B and C.

In order to present the LDBNN algorithm, it is necessary first to introduce some formal notions. First of all, let us consider the function $NN\colon U \times \mathbb{R} \to 2^T$. This function maps a target instance $i \in U$ and a value $k \in \mathbb{R}$ to a subset of T that includes the k nearest neighbors of i in each class of the dataset. That is:

$$NN(i,k) = \bigcup_{l \in L} nn(i,l,k) \tag{4}$$

Where $nn\colon U \times L \times \mathbb{R} \to 2^T$ is a function that maps a given target instance $i \in U$, a given class label $l \in L$ and a given value $k \in \mathbb{R}$ to a subset of T that includes the k nearest neighbors of i within $c(l)$. That is:

$$nn(i,l,k) = \{x | x \text{ is one of the } k \text{ nearest neighbors of } i \text{ within } c(l)\} \tag{5}$$

Another notion that is important in the context of LDBNN is the *similarity-based weight*.

Definition 1. The *similarity-based weight* of a given class label $l \in L$, regarding a given target instance $i \in U$, which should be classified, and a given value $k \in \mathbb{R}$, represents the degree to which the k nearest neighbors of i within the class l are similar to i. Thus, it can be viewed as a measure of the density of the region around i within the class l. The similarity-based weight is given by the function $sw\colon L \times U \times \mathbb{R} \rightarrow \mathbb{R}$, such that:

$$sw(l,i,k) = \sum_{x \in nn(i,l,k)} \frac{1}{1+d(x,i)} \qquad (6)$$

Notice that the greater is the *similarity-based weight* of a given class l, the closer are the k nearest neighbors of l to the target instance i and the denser is the region around i within the class l.

The LDBNN also adopts the notion of *neighbor density*, which was inspired in the notion of density adopted in (Bai et al., 2012; Carbonera and Abel, 2015; Carbonera and Abel, 2016; Carbonera, 2017).

Definition 2. The *neighbor density* of a given neighbor $n \in NN(i,k)$, of a given target instance $i \in T$ and considering a value $k \in \mathbb{R}$, measures the density of n within the region occupied by all the neighbors of i, in $NN(i,k)$. If the density of n is high, this means that n is very similar to the other neighbors of i. The neighbor density is given by the function $nd\colon T \times U \times \mathbb{R} \rightarrow \mathbb{R}$, such that:

$$nd(n,i,k) = \frac{\sum_{x \in NN(i,k)} \frac{1}{1+d(x,n)}}{|NN(i,k)|} \qquad (7)$$

With the notion of *neighbor density*, we can define the notion of *class density*.

Definition 3. The *class density* of a given class $l \in L$, regarding a target instance $i \in T$ and considering a value $k \in \mathbb{R}$, measures the density of the region occupied by the k nearest neighbors of i within the class l. It is given by the function $cd\colon L \times T \times \mathbb{R} \rightarrow \mathbb{R}$, such that:

$$cd(l,i,k) = \frac{\sum_{x \in nn(i,l,k)} nd(x,i,k)}{|nn(i,l,k)|} \qquad (8)$$

The *class density* allows us to define the notion of *density weight*.

Definition 4. The *density weight* of a given class $l \in L$, regarding a target instance $i \in T$ and considering a value $k \in \mathbb{R}$, measures how dense the class l is in comparison with the other classes, in the region occupied by the neighbors of i, in $NN(i,k)$. This notion is captured by the function $dw\colon L \times T \times \mathbb{R} \rightarrow \mathbb{R}$, such that:

$$dw(l,i,k) = \frac{cd(l,i,k)}{\sum_{y \in L} cd(y,i,k)} \qquad (9)$$

Finally, at this point, we can define the notion of *classification weight*.

Definition 5. The *classification weight* of a given class $l \in L$, regarding a target instance $i \in T$ and considering a value $k \in \mathbb{R}$, measures the degree to which i belongs to the class l. This is given by the function $cw\colon L \times T \times \mathbb{R} \rightarrow \mathbb{R}$, such that:

$$cw(l,i,k) = dw(l,i,k) \times sw(l,i,k) \qquad (10)$$

Thus, the *classification weight* of a given class is basically the multiplication of its *similarity weight* and its *density weight*. According to this, the higher are the *similarity weight* and the *density weight*, the higher is the *classification weight*.

Considering to the previously presented notions, the LDBNN can be viewed as a function $LDBNN\colon U \times \mathbb{R} \rightarrow L$, which maps a target instance $i \in U$ and a value $k \in \mathbb{R}$ to the class label $l \in L$ that classifies i, such that:

$$LDBNN(i,k) = l | l \in L \wedge \forall y \in L : cw(y,i,k) \leq cw(l,i,k) \qquad (11)$$

Thus, LDBNN identifies the class $l \in L$ that has the higher *classification weight* for the target instance $i \in U$ and a value $k \in \mathbb{R}$. The algorithm 1 provides an implementation of this strategy.

5 EXPERIMENTS

In our evaluation process, we have compared the LDBNN algorithm with 5 important instance-based classification algorithms provided by the literature: KNN, WKNN, DWKNN, LMKNN and LMPNN. We considered 16 well-known datasets: Audiology, Breast cancer, Cars, Diabetes, E. Coli, Glass, Ionosphere, Iris, Letter, Mushroom, Promoters, Segment, Soybean[1], Splice, Voting, Zoo. All datasets were obtained from the UCI Machine Learning Repository[2]. In Table 1, we present the details of the datasets that were used.

For evaluating the *accuracy* of the algorithms, we adopted a *leave-one-out* schema. Thus, for a given dataset and classification algorithm, each instance was classified using the remaining instances of the dataset as the training set. At the end, we computed the accuracy of the algorithm in the considered dataset as the ratio between the number of correctly classified instances and the number of instances in the whole dataset. Since that the algorithms KNN, WKNN, DWKNN, LMKNN and LMPNN need a parameter k, in our experiments we followed the strategy adopted by (Gou et al., 2014), which considered the values of k in the interval $[1,15]$. Thus, we carried out a complete leave-one-out evaluation process for

[1]This dataset combines the large soybean dataset and its corresponding test dataset.

[2]http://archive.ics.uci.edu/ml/

Algorithm 1: LDBNN (Local Density-based Nearest Neighbors) algorithm.

```
Input: A target instance $i$, the training set $T$
       and the number $k$ of neighbors.
Output: A class label $cl$ that classify $i$, such
        that $cl \in L$.
begin
    $MAX \leftarrow 0$;
    $class \leftarrow null$;
    $all \leftarrow \emptyset$; foreach $l \in L$ do
        $KNN_l \leftarrow nn(i,l,k)$;
        $all \leftarrow all \bigcup KNN_l$;
        $sw_l \leftarrow 0$;
        foreach $x \in KNN_l$ do
            $sw_l \leftarrow sw_l + \frac{1}{1+d(x,i)}$;
        $sw_l \leftarrow \frac{sw_l}{k}$;
    $totalD \leftarrow 0$;
    foreach $l \in L$ do
        $cD_l \leftarrow 0$;
        foreach $x \in KNN_l$ do
            $nD \leftarrow 0$;
            foreach $y \in all$ do
                $nD \leftarrow nD + \frac{1}{1+d(x,y)}$;
            $nD \leftarrow \frac{nD}{|all|}$;
            $cD_l \leftarrow classD_l + nD$;
        $totalD \leftarrow totalD + cD_l$;
    foreach $l \in L$ do
        $dw_l \leftarrow \frac{cD_l}{totalD}$;
        $cW_l \leftarrow sw_l \times dw_l$;
        if $cW_l > MAX$ then
            $MAX \leftarrow cW_l$;
            $class \leftarrow l$;
    return $class$;
```

each value of k. This process produces 15 accuracy values for each combination of algorithm and dataset (one accuracy value for each value of k). Finally, from this set of values, we select the greatest accuracy value obtained by each algorithm in each dataset. The accuracies obtained during our experiments are presented in Table 2.

In this evaluation process, we adopted the following distance function $d\colon T \times T \to \mathbb{R}$:

$$d(x,y) = \sum_{j=1}^{m} \theta_j(x,y) \tag{12}$$

where

$$\theta_j(x,y) = \begin{cases} \alpha(x_j,y_j), & \text{if } j \text{ is a categorical feature} \\ |x_j - y_j|, & \text{if } j \text{ is a numerical feature} \end{cases} \tag{13}$$

Table 1: Details of the datasets used in the evaluation process.

Dataset	Instances	Attributes	Classes
Audiology	226	70	24
Breast cancer	286	10	2
Cars	1728	6	4
Diabetes	768	9	2
E. Coli	336	8	8
Glass	214	10	7
Ionosphere	351	35	2
Iris	150	5	3
Letter	20000	17	26
Mushroom	8124	23	2
Promoters	106	58	2
Segment	2310	20	7
Soybean	683	36	19
Splice	3190	61	3
Voting	435	17	2
Zoo	101	18	7

where

$$\alpha(x_j,y_j) = \begin{cases} 1, & \text{if } x_j \neq y_j \\ 0, & \text{if } x_{yj} = y_j \end{cases} \tag{14}$$

This distance function was also adopted in (Carbonera and Abel, 2015) for dealing with data that is described by both numerical and categorical values.

Table 2 shows that the LDBNN algorithm achieves the highest accuracy in most of the datasets and also achieves the highest average accuracy. Also, although the results achieved by LDBNN are similar to the ones achieved by LMPNN in some datasets, it is important to notice that in some datasets (Audiology and Breast cancer, for example) the performance of LDBNN is significantly better than the performance of LMPNN. Besides that, only in the *Segment* dataset the performance of LMPNN was higher than the performance of LDBNN, and the difference between the accuracy achieved by both the algorithms is very small. This shows that LDBNN and LMPNN have different profiles of performance and suggests that the notions that underly the LDBNN algorithm can be used for developing better algorithms in the future.

6 CONCLUSION

In this paper, we propose a novel knn-based algorithm called LDBNN (*Local Density-based nearest neighbors*). This algorithm considers that a target instance should be classified in a class whose the k nearest neighbors constitute a dense region, where the neighbors are near to each other and also near to the target instance. This strategy makes the algorithm more resistant to the effects of noise in the dataset, since noisy

Table 2: The accuracy of each algorithm in each dataset.

Algorithm	KNN	WKNN	DWKNN	LMKNN	LMPNN	LDBNN	Average
Audiology	0.73	0.73	0.76	0.73	0.73	**0.80**	0.75
Breast cancer	0.73	0.73	0.74	0.66	0.71	**0.76**	0.72
Cars	0.70	0.70	**0.95**	0.93	0.93	**0.95**	0.86
Diabetes	0.69	0.69	0.71	0.73	0.72	**0.75**	0.71
E. Coli	0.82	0.82	0.83	**0.87**	0.86	**0.87**	0.84
Glass	0.73	0.73	0.75	0.75	**0.76**	**0.76**	0.75
Ionosphere	0.91	0.91	0.91	**0.92**	0.91	0.91	0.91
Iris	0.95	0.95	0.95	**0.97**	0.96	0.96	0.96
Letter	0.96	0.96	0.97	**0.97**	**0.97**	**0.97**	0.97
Mushroom	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	1.00
Promoters	0.78	0.78	0.79	0.80	0.86	**0.89**	0.82
Segment	0.97	0.97	0.97	0.97	**0.98**	0.97	0.97
Soybean	0.92	0.92	0.92	0.91	**0.93**	**0.93**	0.92
Splice	0.79	0.79	0.80	0.75	0.84	**0.87**	0.81
Voting	0.92	0.92	0.93	0.93	**0.94**	**0.94**	0.93
Zoo	0.96	0.96	0.96	**0.98**	**0.98**	**0.98**	0.97
Average	0.85	0.85	0.87	0.87	0.88	**0.89**	0.82

instances, are sparsely distributed, in general.

The experiments show that LDBNN achieves the highest accuracy in most of the considered datasets and the highest average accuracy. Also, the LDBNN algorithm is the only algorithm whose accuracy is higher than the average in all datasets. Although, in some datasets, the LDBNN achieves accuracy rates that are similar to those achieved by other algorithms (such as LMPNN), in some datasets it provides much higher accuracy rates. This suggests that the concepts underlying the LDBNN algorithm are powerful notions that should be investigated in the future for developing better classification algorithms.

In the future, we plan to investigate how to combine the LDBNN algorithm with other algorithms that achieve a significant performance, such as the LMPNN algorithm.

REFERENCES

Aggarwal, C. C. and Reddy, C. K., editors (2014). *Data Clustering: Algorithms and Applications*. CRC Press.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

Bai, L., Liang, J., Dang, C., and Cao, F. (2012). A cluster centers initialization method for clustering categorical data. *Expert Systems with Applications*, 39(9):8022–8029.

Carbonera, J. L. (2017). An efficient approach for instance selection. In *International conference on big data analytics and knowledge discovery*, pages 228–243. Springer.

Carbonera, J. L. and Abel, M. (2015). A density-based approach for instance selection. In *IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 768–774. IEEE.

Carbonera, J. L. and Abel, M. (2016). A novel density-based approach for instance selection. In *IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 549–556. IEEE.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE TRansactions on Information Theory*, 13(1):21–27.

Deekshatulu, B., Chandra, P., et al. (2013). Classification of heart disease using k-nearest neighbor and genetic algorithm. *Procedia Technology*, 10:85–94.

Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, (4):325–327.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Gou, J., Xiong, T., and Kuang, Y. (2011). A novel weighted voting for k-nearest neighbor rule. *Journal of Computers*, 6(5):833–840.

Gou, J., Zhan, Y., Rao, Y., Shen, X., Wang, X., and He, W. (2014). Improved pseudo nearest neighbor classification. *Knowledge-Based Systems*, 70:361–375.

Imandoust, S. B. and Bolandraftar, M. (2013). Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background. *International Journal of Engineering Research and Applications*, 3(5):605–610.

Li, L., Zhang, Y., and Zhao, Y. (2008). k-nearest neighbors for automated classification of celestial objects. *Science in China Series G: Physics, Mechanics and Astronomy*, 51(7):916–922.

Mitani, Y. and Hamamoto, Y. (2006). A local mean-based nonparametric classifier. *Pattern Recognition Letters*, 27(10):1151–1159.

Murty, M. N. and Devi, V. S. (2011). Nearest neighbour based classifiers. In *Pattern Recognition*, pages 48–85. Springer.

Ramírez, J. F., Fuentes, O., and Gulati, R. K. (2001). Prediction of stellar atmospheric parameters using instance-based machine learning and genetic algorithms. *Experimental Astronomy*, 12(3):163–178.

Reese, H., Granqvist-Pahlén, T., Egberth, M., Nilsson, M., and Olsson, H. (2005). Automated estimation of forest parameters for sweden using landsat data and the knn algorithm. In *Proceedings, 31st International Symposium on Remote Sensing of Environment*.

Sarkar, M. and Leong, T.-Y. (2000). Application of k-nearest neighbors algorithm on breast cancer diagnosis problem. In *Proceedings of the AMIA Symposium*, page 759. American Medical Informatics Association.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., et al. (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37.

Wu, Y. and Ai, X. (2008). Face detection in color images using adaboost algorithm based on skin color information. *International Workshop on Knowledge Discovery and Data Mining*, 0:339–342.

Yong, Z., Youwen, L., and Shixiong, X. (2009). An improved knn text classification algorithm based on clustering. *Journal of computers*, 4(3):230–237.