Soft Computing manuscript No.
(will be inserted by the editor)

A fuzzy k-nearest neighbor classifier to deal with imperfect data

Jose M. Cadenas · M. Carmen Garrido · Raquel Martínez · Enrique Muñoz · Piero P. Bonissone

Abstract The k-nearest neighbors method (kNN) is a non-parametric, instance-based method used for regression and classification. To classify a new instance, the kNN method computes its "k" nearest neighbors and generates a class value from them. Usually, this method requires that the information available in the datasets be precise and accurate, except for the existence of missing values. However, data imperfection is inevitable when dealing with real-world scenarios. In this paper, we present the kNN$_{imp}$ classifier, a k-nearest neighbors method to perform classification from datasets with imperfect value. The importance of each neighbor in the output decision is based on relative distance and its degree of imperfection. Furthermore, by using external parameters, the classifier enables us to define the maximum allowed imperfection, and to decide if the final output could be derived solely from the greatest weight class (the best class) or from the best class and a weighted combination of the closest classes to the best one.

To test the proposed method, we performed several experiments with both synthetic and real-world datasets with imperfect data. The results, validated through statistical tests, show that the kNN$_{imp}$ classifier is robust when working with imperfect data and maintains a good performance when compared with other methods in the literature, applied to datasets with or without imperfection.

Keywords k-nearest neighbors · Classification · Imperfect Data · Distance/Dissimilarity Measures · Combination Methods

1 Introduction

Data mining is a main phase of Intelligent Data Analysis (IDA). This phase has as fundamental aim to find comprehensible models from data. It can be developed using several tasks being one of the most challenging tasks, the classification task [12, 19, 21]. There are many techniques developed to tackle the classification problems, however, most of them do not take into account the imperfection contained in

Jose M. Cadenas · M. Carmen Garrido
Dept. of Information Engineering and Communication. University of Murcia. Murcia, Spain
E-mail: jcadenas@um.es

M.C. Garrido
E-mail: carmengarrido@um.es

Raquel Martínez
Dept. of Computer Engineering, Catholic University of San Antonio. Murcia, Spain
E-mail: rmartinez@ucam.edu

Enrique Muñoz
Dept. of Computer Science, Università degli Studi di Milano. Crema, Italy
E-mail: enrique.munoz@unimi.it

Piero P. Bonissone
Piero P. Bosissone Analytics (independent consultant), LLC. San Diego, CA, USA
E-mail: bonissone@gmail.com

data. This imperfection can appear for example due to errors during data obtaining or due to the high cost to obtain precise information, between other factors.

When imperfection in the data is insignificant, it is usually removed, as it is widely accepted by researchers that the amount of information that is lost when you delete it is very small. However, imperfection in the data is increasingly common and therefore deleting information could suppose the loss of relevant information. Hence the different forms of imperfection inherent in the real world problems need to be handled properly, in order to solve some practical problems without losing information [45, 47,48,58]. The imperfect data can be represented by fuzzy concepts, crisp subset, fuzzy subset, interval concepts, etc.

One of the difficulties to deal with imperfect information problems is that existing techniques do not manage imperfect information explicitly; therefore the idea is to propose new techniques or to adapt existing techniques to be able to manage this kind of information without any transformation of the data and avoiding losing relevant information. There are already some initiatives on this subject appearing in different papers, which is going to be exposed in detail in Section 2.

In this paper, we propose to adapt a well-known technique, the fuzzy k-nearest neighbors classifier. The classic fuzzy k-nearest neighbors technique requires datasets with precise and accurate information, except for the existence of missing values and we propose a fuzzy k-nearest neighbors based classifier denoted by kNN$_{imp}$ which deals with imperfect data. The classifier assigns a higher weight to those neighbors with less imperfection and nearest.

This work is organized as follows: In Section 2, we analyze the state of the art in methods dealing with imperfect information. In Section 3 the major elements that constitute the fuzzy k-nearest neighbor classifier kNN$_{imp}$ is exposed. Section 4 describes our experimental framework, including datasets, setting parameters, classification accuracy and statistical procedure. In Section 5, some experiments are carried out to analyze the robustness and efficacy of the method proposed with datasets with/without imperfect values. Finally, our conclusions are presented in Section 6.

2 Imperfect data sources in Data Mining

Since the introduction by Zadeh of Fuzzy Sets Theory (FST) in 1965 [63], there have been many areas in which this theory has been applied, one of which is the area of Artificial Intelligence. More specifically, many works have used this theory in the process of IDA, which is defined as the process of extracting useful, comprehensible, hitherto unknown knowledge from huge amounts of data stored in a variety of formats [61].

IDA process consists of a set of phases: data acquisition, preprocessing, data mining, evaluation-interpretation and dissemination.

In the data acquisition, the most relevant sources of information are selected and converted to the same format. The preprocessing is one of the most complex phases of the IDA and aims to transform the original data sources in a dataset prepared for use by a specific technique in the next phase. This dataset is named minable view. The more requirements a technique imposes on the types of values that it can deal, the greater the effort to make at this stage is. Data mining is a phase that aims to find intelligent models by applying a technique or specific algorithm to the dataset we have called minable view. In the evaluation-interpretation phase, models obtained from the previous phase are evaluated and interpreted, returning to previous phases if necessary. Finally, in the dissemination phase, the new knowledge is used.

In the data mining phase, FST has been applied at what we can consider two levels:

- Only at the level of generated models: Techniques that generate models described by fuzzy logic elements which are more interpretable. So we can find elements of fuzzy logic in rule-based systems, techniques based on k-nearest neighbors, decision trees, clustering and support vector machines.
  In 1971 Zadeh proposed the design of rules IF-THEN using linguistic variables that can be provided by a group of experts or obtained through data mining techniques. So, among others, in [5] a set of fuzzy rules is obtained using a method based on genetic programming, in [20] a set of fuzzy rules is obtained in unbalanced problems using a genetic selection process of rules, in [28] different weights are assigned to a set of fuzzy rules using heuristic methods, in [43] An initial set of fuzzy rules is constructed by clustering and then are optimized using a neuro-fuzzy learning algorithm.

Among the fuzzy versions of the k-nearest neighbors rule we can highlight works that assign fuzzy memberships of each instance to each class, use fuzzy distance measures, use different ways of combining the votes of neighbors, etc. A complete review of these methods is carried out in [15].
Fuzzy decision trees have also been designed as the proposed in [44] that obtains in each node to split, the best fuzzy partition of the best attribute at that node and fuzzy ensembles such as the proposed in [13] where a fuzzy decision tree ensemble is constructed from a non-fuzzy tree construction algorithm that subsequently is transformed to fuzzy.
With the aim to construct data partitions that allow an instance belongs to more than one partition, fuzzy clustering algorithms have been developed such as the fuzzy C-means proposed in [6]. Different versions of this algorithm are found in [25] to extend it to nominal data, in [35] to deal with missing values through intervals defined by the nearest neighbors or in [57] where it is used in order to design an algorithm of hierarchical fuzzy clusters.
Also, fuzzy versions of support vector machines have been designed. So, in [37] a degree of membership to each class is assigned to each instance, allowing to each one to contribute differently in the learning of the decision surface. In [27] a truncated polyhedral pyramidal membership function is used to allow the classification of instances that make positive more than two decision functions in problems with more than two classes.

– At minable view level: Techniques that besides incorporating FST elements, support input data described through fuzzy logic. In this case, the techniques allow us that the available data are composed of attributes described by values that are not as "perfect" as desirable. This generates the following advantages:1) techniques can interpret the imprecision and uncertainty expressed in the data and generate robust models to these types of information without transforming the true nature of them; 2) data preprocessing is simplified by not carrying out these transformations (substitutions, imputations, deleting data, ...); and 3) the minable view contains a greater number of instances because the imprecise and uncertain data are not discarded.
In general, significant efforts are being carried out to incorporate the treatment of imprecise and uncertain data into data mining techniques using the FST. Thus we can find works that incorporate the treatment of fuzzy values. There are fuzzy decision trees based on a fuzzy partition of numerical attributes of the problem. This partition is used in the test of nodes based on numerical attributes as in [30,34]. Fuzzy partitions of numerical attributes are also used in the construction of fuzzy ensembles to incorporate input fuzzy values. This approach is used in [31] where to select the test of each node, the set of the best attributes for partitioning that node is used, in [7,9] where the classic ensemble random forest is extended to a fuzzy random forest or in [39] where one fuzzy ensemble for each class value of the problem is constructed. In [41] a fuzzy version of multilayer perceptron is presented which performs the learning from fuzzy values. In [24] the treatment of fuzzy values is performed incorporating the Demspter-Shafer Evidence Theory to a mixture model. In [46] a genetic classifier based on fuzzy rules is obtained from data described with fuzzy values. In [49] and [50] Adaboost and FURIA algorithms are extended in orden to obtain fuzzy rules from fuzzy values respectively. In [51] an algorithm to obtain a set of fuzzy association rules from a fuzzy partition previously obtained is proposed.
As particular cases of fuzzy values, some works deal with values expressed by intervals as in [7,9,24, 46,49–51,34].
On the other hand, the set of techniques that allow the existence of missing values is considerable. We highlight only a few that allow the treatment of some other type of imperfect information as [7,9,24, 30,31] or as in [34], where missing values are only allowed in the classification phase.
Finally, there is a considerable set of techniques that have considered the possibility that an instance has more than one associated class value (multi-valued class), but few extend this possibility to other nominal attributes of a problem (multi-valued attributes). So, among the first we can find works as [46] where class may be defined by a crisp set, or [62] where a fuzzy k-nearest neighbor method is used where an instance can belong to more than one class with several degrees. In [38] we can find a comparison of this kind of methods. Among the latter we can highlight [7,9] where any nominal value can be expressed through crisp/fuzzy sets, [24] where nominal values are expressed by both probability and possibility distributions using the the Demspter-Shafer Evidence Theory as framework

and [11] where a modification of the decision tree algorithm C4.5 is done to deal with multi-valued attributes.

In this work we propose a technique based on the k-nearest neighbors rule to classify incorporating the FST at minable view level, allowing that the input data contain fuzzy and interval values in numerical attributes and values expressed by crisp/fuzzy sets in nominal attributes (multi-valued attributes), besides allowing the existence of missing values in either types of attributes. The output provided by the classifier is also expressed by a crisp/fuzzy set.

3 A fuzzy k-nearest neighbors classifier to cope imperfect data

3.1 Introduction

Considering the taxonomy introduced in [15], kNN$_{imp}$ classifier falls in the following cathegories:

- Fuzzy sets based method. kNN$_{imp}$ classifier represents imperfection in data using fuzzy sets.
- Dependent on the k value, because it uses k to calculate the neighbors used to classify.
- Use of distance. kNN$_{imp}$ classifier weights neighbors' decisions using a fuzzy distance/dissimilarity measure.
- Use of fuzzy/crisp weight, because it weights the decision depending on the degree of imperfection of the samples.

3.2 Notation and types of imperfect values

Let us consider a set of instances E. Each instance $\mathbf{x}$ is characterized by a number of n attributes in a vector $(x_1, x_2, \ldots, x_n)$, where the n-th attribute represents the class. The domains of each attribute, $\Omega_{x_1}, \Omega_{x_2}, \ldots, \Omega_{x_{n-1}}$, can be numerical or nominal, while the domain of the class Ω_{x_n} can take the values $\{\omega_1, \omega_2, \ldots, \omega_I\}$.

kNN$_{imp}$ classifier represents numerical attributes as fuzzy sets with a trapezoidal fuzzy membership function [4] $\mu(x)$ defined by a quadruple (a, b, c, d):

$$\mu(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \leq x < b \\ 1 & b \leq x < c \\ \frac{d-x}{d-c} & c \leq x < d \\ 0 & x \geq d \end{cases}$$

In particular, the proposed strategy can deal with the following types of numerical attributes:

- Crisp values. This kind of attributes includes data with minimal or no imperfection. kNN$_{imp}$ classifier represents the crisp value a as the quadruple (a, a, a, a).
- Interval values. Intervals analysis has been used to deal with uncertainty and imprecision in data [42]. In this case, a classical set substitutes the crisp value. However, the membership degree is binary. kNN$_{imp}$ classifier models the interval $[a, b]$ as the quadruple (a, a, b, b).
- Fuzzy values. Fuzzy sets are a common tool to represent both uncertainty and imprecision [8]. Compared with intervals, fuzzy sets represent the concept of approximate numbers by assigning a membership degree $\mu \in [0, 1]$. kNN$_{imp}$ classifier uses the trapezoidal fuzzy membership function defined above to represent fuzzy sets.
- Missing values. This kind of data includes pieces of information that are unknown. kNN$_{imp}$ classifier models missing data belonging to attribute x_i using the quadruple $(min_i, min_i, max_i, max_i)$, where min_i and max_i are, respectively, the minimum and maximum values of Ω_{x_i} included in the training dataset.

kNN$_{imp}$ classifier models nominal attributes x_i as fuzzy subsets $\{\mu(h_1)/h_1, \ldots, \mu(h_s)/h_s\}$, where h_j is a domain value in Ω_i and $\exists h_k \in \Omega_i : \mu(h_k) = 1$. kNN$_{imp}$ classifier can manage the following types of nominal attributes:

- Crisp values. kNN$_{imp}$ classifier represents the nominal crisp value h_j as the fuzzy subset $1/h_j$.
- Crisp subset values. This kind of data considers more than a possible nominal value. They are represented as $\{1/h_1, \ldots, 1/h_s\}$.
- Fuzzy subset values. This domain considers more than one nominal value with a membership value $\mu \in [0, 1]$. They are represented using the notation introduced above.
- Missing nominal values. This kind of unknown values are represented using a fuzzy subset that contains all possible values with membership degree equals to 1.

3.3 The inference process of the kNN$_{imp}$ classifier

Algorithm 1 describes the process followed by kNN$_{imp}$ classifier to infer the class of a new instance $\mathbf{z}$. In the following sections we will introduce all the functions and parameters that determine the classification process of kNN$_{imp}$.

As Algorithm 1 shows, kNN$_{imp}$ classifier computes the set $KIMP_z$ that contains the k instances $\mathbf{x^j} \in E$ which are nearest to z according to the measure $d_{imp}(\mathbf{x^j}, \mathbf{z})$. Then, for each instance $\mathbf{x^j} \in KIMP_z$, two weights are calculated depending on its degree of imperfection ($p(\cdot)$) and its distance to $\mathbf{z}$ ($q(\cdot)$). Furthermore, the overall degree of imperfection in $KIMP_z$ is measured, if it is too high, the classification is not performed. To establish the maximum degree of imperfection, kNN$_{imp}$ uses the parameter U_I. If $KIMP_z$ passes the imperfection check, the functions $AggreN$ and $AggreF$ obtain the set of possible weighted classes taking into account the k-nearest neighbors. The class with the highest score is chosen as output, together with other classes whose score is similar to the highest. To assess if a class should be included in the final output, kNN$_{imp}$ uses the threshold U_D.

Algorithm 1: kNN$_{imp}$ classifier - k Nearest Neighbor from imperfect data

Input Dataset E, Instance to classify $\mathbf{z}$, Value k; $1 \leq k \leq |E|$, Values U_D and U_I ($U_D, U_I \in [0, 1]$)
Let $KIMP_z$ be the set of the k-nearest instances of $\mathbf{z}$ according to $d_{imp}(\cdot, \cdot)$
Calculate imperfection weight ($p(\mathbf{x^j})$) and distance weight ($q(\mathbf{x^j})$) for all $\mathbf{x^j} \in KIMP_z$
if degree of imperfection of $KIMP_z$ is smaller or equal than U_I **then**
  Aggregate the information of each neighbor in order to obtain possible class values for the instance $\mathbf{z}$ using $AggreN$ and $AggreF$ functions
  Calculate the set of output classes z_n using U_D
  Output z_n
else
  Output Classification is not performed
end if

3.4 Contribution of neighbors to the classification and control parameters

In this section, the different elements that are part of the kNN$_{imp}$ classifier process (Algorithm 1) are defined. The Section 3.4.1 defines the distance/dissimilarity measure used $d_{imp}(\cdot, \cdot)$, the Section 3.4.2 defines the weights in the classification from different neighbors selected ($q(\cdot)$ and $p(\cdot)$). Finally, the kNN$_{imp}$ classifier incorporates two control parameters for decision-making, that are defined in the Sections 3.4.3 and 3.4.4. The first one allows us to make a decision about the permitted imperfection and the second one on the output classes.

3.4.1 Distance/Dissimilarity measure

In order to calculate the nearest neighbors necessary for the classification, we need to define a measure, which computes the distance between two instances and can work with/without imperfect data coming from numerical and nominal attributes. Let us define the function $d_{imp}(\mathbf{x}, \mathbf{x}')$, between two instances as:

$$d_{imp}(\mathbf{x}, \mathbf{x}') = \sqrt{\frac{\sum_{i=1}^{n-1} f(x_i, x_i')^2}{n-1}} \quad \text{with } f(x_i, x_i') = \begin{cases} f_1(x_i, x_i') & \text{if } \Omega_{x_i} \text{ is numerical} \\ f_2(x_i, x_i') & \text{if } \Omega_{x_i} \text{ is nominal} \end{cases}$$

$d_{imp}(\mathbf{x}, \mathbf{x}')$ is a heterogeneous function defined from different functions, $f_1(\cdot, \cdot)$ and $f_2(\cdot, \cdot)$, on different kinds of attributes where $f_1(\cdot, \cdot)$ and $f_2(\cdot, \cdot)$ are normalized fuzzy distance or dissimilarity measures.

kNN$_{imp}$ classifier uses $d_{imp}(\cdot, \cdot)$ to compute the set $KIMP_z$ that contains the k nearest neighbors of a given instance $\mathbf{z}$. Afterward, the information contained in $KIMP_z$ is exploited to produce the classification of $\mathbf{z}$. To do that, kNN$_{imp}$ classifier uses two different weights introduced below.

3.4.2 Assigning weights to the nearest neighbors

Weights based on distance.- kNN$_{imp}$ classifier considers that those neighbors in $KIMP_z$ that are further from $\mathbf{z}$ should have less influence in the final decision. To reflect this fact, the proposed strategy introduces the weight as:

$$q(\mathbf{x}) = 1 - d_{imp}(\mathbf{x}, \mathbf{z}) \quad \text{with } \mathbf{x} \in KIMP_z$$

Weights based on imperfection.- The instances in $KIMP_z$ can contain imperfect data, which has to be considered during the classification of $\mathbf{z}$. kNN$_{imp}$ classifier calculates a weight for each neighbor taking into account a measure of its degree of imperfection. In this way, a neighbor that contains a high degree of imperfection has less influence in the result than another one with a lower degree. We define this weight as:

$$p(\mathbf{x}) = 1 - imp(\mathbf{x}) \quad \text{with } \mathbf{x} \in KIMP_z$$

and $imp(\cdot) : E \rightarrow [0, 1]$ as:

$$imp(\mathbf{x}) = \left(\sum_{i=1}^{n} g(x_i) \right) / n$$

where $g(\cdot)$ is a function defined for each attribute x_i as $g : \Omega_{x_i} \rightarrow [0, 1]$. This function measures the imperfection of the value in the attribute x_i.

Some examples of $g(\cdot)$ measures that can serve us to measure the imperfection are the fuzzy entropy functions or the power of fuzzy sets [14, 17, 33][1].

[1] For example, the fuzzy entropy (Ent$(\cdot)$) and the power of fuzzy sets (Pw$(\cdot)$) defined by DeLuca and Termini [14] are the following:

$$Ent(A) = \sum_{a \in A} (\mu(a) log(\mu_A(a)) + (1 - \mu_A(a)) log(1 - \mu_A(a)))$$

$$Pw(A) = \sum_{a \in A} \mu_A(a)$$

where A is a fuzzy set and in the case of continuous fuzzy sets, the sum is understood as an integral.

3.4.3 Controlling the tolerated imperfection degree

We can measure the average imperfection between the nearest neighbors $\mathbf{x^j} \in KIMP_z$, defined as $\overline{imp}(KIMP_z) = \frac{\sum_{\mathbf{x^j} \in KIMP_z} imp(\mathbf{x^j})}{k}$, to decide if there is enough information to perform the classification. As we discussed above, the imperfection value $imp(\cdot)$ is used to weight the importance of $\mathbf{x^j}$ in the classification. Thus, if $\overline{imp}(KIMP_z)$ is very high, we can indicate that kNN$_{imp}$ classifier cannot carry out the classification. To offer this possibility, we have introduced the threshold $0 \leq U_I \leq 1$, which limits $\overline{imp}(KIMP_z)$. When $U_I = 0$, kNN$_{imp}$ only performs the classification if all the neighbors only contain crisp values. On the other hand, when $U_I = 1$ kNN$_{imp}$ can perform the classification regardless of the degree of imperfection of the nearest neighbors.

3.4.4 Controlling the similarity in the output classes

The kNN$_{imp}$ classifier exploits the definition of a similarity value between possible classes, defined as $sim(\omega_M, \omega_i) = \frac{\mu(\omega_M) - \mu(\omega_i)}{\mu(\omega_M)}$, to perform the classification of an instance. The minimum $sim(\omega_M, \omega_i)$ necessary to consider that the classes ω_M and ω_i are possible outputs is controlled by the threshold $0 \leq U_D \leq 1$. Thus, let us assume that ω_c is the class having the highest membership degree $\mu(\omega_c)$ to classify an instance. If there are other classes with very close membership degrees to $\mu(\omega_c)$, we could return all these classes as possible classification of the instance. The role of U_D threshold is to define how close to ω_c must be a class to be considered an output class. When $U_D = 0$, the output class is generally a set that contains a single class (the class with the highest membership degree), although the set can contain more than one class if there is a tie between two or more classes that have the same highest membership degree. Alternatively, if $U_D = t$, kNN$_{imp}$ returns the class with the highest membership degree, and those whose membership degree differ relatively from the highest membership degree at most by "t". For example, we do not want to decide between a class "a" with membership degree $\mu(a) = 0.52$ and a class "b" with membership degree $\mu(b) = 0.48$ because they have very similar membership degrees. If $U_D = 0.1$ then $\frac{0.52 - 0.48}{0.52} = 0.074 < 0.1$ and therefore, the classifier returns both classes {a,b}. Therefore, an output class value of kNN$_{imp}$ can be a crisp value or a crisp set.

3.5 Aggregation methods for classification

As we have commented, the aggregation methods that we define for kNN$_{imp}$ classifier are composed of the two functions $AggreN(\cdot)$ and $AggreF(\cdot)$. These two functions provide high flexibility to kNN$_{imp}$ classifier, allowing choose them according to the classification problem. In the following sections we propose different possible $AggreN(\cdot)$ and $AggreF(\cdot)$ functions. At the end of this section, a clear and simple example is included to show the decision of the different aggregation methods.

3.5.1 Functions to define the aggregation methods

- The $AggreN(\cdot,\cdot,\cdot,\cdot)$ function aggregates the information that each nearest neighbor $\mathbf{x^j}$ provides for each possible class $\omega_h \in \Omega_{x_n}$. Below, several possible definitions are presented:

  - Methods based on simple vote of each neighbor (denoted $SVEN()$). These methods have in common that each neighbor $\mathbf{x^j}$ of $\mathbf{z}$ provides a vote of 1 to the class of $\mathbf{x^j}$ with the highest membership degree (let us remember that it is possible that an instance has more than a class with different membership degrees).
    $AggreN() = SVEN()$ function is defined as follows:

$$SVEN(i, x_n^j, p(\mathbf{x^j}), q(\mathbf{x^j})) = \begin{cases} 1 & \text{if } i = \arg \max\limits_{h,\ h=1,\ldots,I} \mu^j(\omega_h) \\ 0 & \text{otherwise} \end{cases}$$

– Methods based on simple vote weighted by the importance of each neighbor (denoted $WSVEN()$). These methods have in common that each nearest neighbor $\mathbf{x^j}$ selects the class of $\mathbf{x^j}$ with the highest membership degree and assigns to it a weight defined by its importance as neighbor. The importance of $\mathbf{x^j}$ as neighbor is measured according to its distance to $\mathbf{z}$ and its degree of imperfection.
$AggreN() = WSVEN()$ function is defined as follows:

$$WSVEN(i, x_n^j, p(\mathbf{x^j}), q(\mathbf{x^j})) = \begin{cases} p(\mathbf{x^j}) \cdot q(\mathbf{x^j}) & \text{if } i = \arg \max_{h,\ h=1,\ldots,I} \mu^j(\omega_h) \\ 0 & \text{otherwise} \end{cases}$$

– Methods based on a vote weighted by the membership degree to each class and importance of each neighbor (denoted $WCVEN()$). These methods have in common that each nearest neighbor $\mathbf{x^j}$ provides a weighted vote to a subset of classes depending on its membership degree to each class value and its importance as neighbor. Again, the importance of $\mathbf{x^j}$ as neighbor is measured according to its distance to $\mathbf{z}$ and its degree of imperfection.
$AggreN() = WCVEN()$ function is defined as follows:

$$WCVEN(i, x_n^j, p(\mathbf{x^j}), q(\mathbf{x^j})) = \mu^j(\omega_i) \cdot p(\mathbf{x^j}) \cdot q(\mathbf{x^j})$$

- The $AggreF(\cdot)$ function decides the output class by aggregating the information provided by each neighbor using the $AggreN(\cdot)$ function. $AggreF(\cdot)$ function takes as input a matrix of size $k \times I$ containing the information provided by each neighbor for each class value. We denote this matrix as INN. We propose two definitions for $AggreF(\cdot)$, $SV(INN)$ and $CV(INN)$, which are introduced in Algorithm 2 and Algorithm 3, respectively. $SV(INN)$ provides the most voted class between the nearest neighbors (it could return more than one class in case of tie), whereas $CV(INN)$ provides all possible classes with degrees obtained from the average information from neighbors.

Algorithm 2: SV(INN)

Input INN

$FS\omega = \{\}$

for $i = 1, \ldots, I$ **do** $AINN(i) = \sum_{j=1}^{k} INN(j, i)$

if $i = \arg \max_{h=1,\ldots,I} AINN(h)$ **then** $max_{\mu(\omega)} = \mu(\omega_i)$

for $i = 1, \ldots, I$ **do**

  if $\mu(\omega_i) = max_{\mu(\omega)}$ **then** $FS\omega = FS\omega \cup \{(\mu(\omega_i) = 1)/\omega_i\}$

Output $FS\omega$

3.5.2 Defining the aggregation methods to kNN$_{imp}$ classifier

We describe several methods to combine the different definitions of $AggreN(\cdot)$ and $AggreF(\cdot)$ functions.

- Methods based on simple vote of each neighbor.
  These methods have in common that each nearest neighbor $\mathbf{x^j}$ provides a vote of 1 to the class value with the highest membership degree.
  – **SM$_{\mathbf{SV}}$** method defined by $AggreN() = SVEN()$ and $AggreF() = SV()$.
  – **SM$_{\mathbf{CV}}$** method defined by $AggreN() = SVEN()$ and $AggreF() = CV()$.
- Methods based on simple vote weighted by the importance of each neighbor.
  These methods have in common that each nearest neighbor $\mathbf{x^j}$ selects the class of $\mathbf{x^j}$ with the highest membership degree and assigns to it a weight defined by its importance as neighbor.

Algorithm 3: CV(INN)

Input INN

$FS\omega = \{\}$

for $i = 1, \ldots, I$ **do** $AINN(i) = \sum_{j=1}^{k} INN(j, i)$

for $i = 1, \ldots, I$ **do** $\mu(\omega_i) = \frac{AINN(i)}{\sum_{h=1}^{I} AINN(h)}$

for $i = 1, \ldots, I$ **do**

  if $\mu(\omega_i) > 0$ **then** $FS\omega = FS\omega \cup \{\mu(\omega_i)/\omega_i\}$

Output $FS\omega$

  - **WSM$_{\mathbf{SV}}$** method defined by $AggreN() = WSVEN()$ and $AggreF() = SV()$.
  - **WSM$_{\mathbf{CV}}$** method defined by $AggreN() = WSVEN()$ and $AggreF() = CV()$.
- Methods based on a vote weighted by the membership degree to each class and importance of each neighbor.
  These methods have in common that each nearest neighbor $\mathbf{x^j}$ provides a weighted vote to a subset of classes depending on its membership degree to each class value and its importance as neighbor.
  - **WM$_{\mathbf{SV}}$** method is defined by $AggreN() = WCVEN()$ and $AggreF() = SV()$.
  - **WM$_{\mathbf{CV}}$** method is defined by $AggreN() = WCVEN()$ and $AggreF() = CV()$.

3.5.3 An illustrative example

To show the definition of the different aggregation methods, we suppose a problem with two possible class values, $\Omega_{x_n} = \{A, B\}$, where for a new instance to classify $\mathbf{z}$, the two nearest neighbors ($k = 2$) are obtained, $\mathbf{x^1}$ and $\mathbf{x^2}$. Table 1 shows the information of these two neighbors: the weights $p(\cdot)$, $q(\cdot)$ and the value in the class attribute.

Table 1 Information from the two neighbors of $\mathbf{z}$

Neighbors of z	$p(\cdot)$	$q(\cdot)$	x_n^i (Neighbor' class)
$\mathbf{x^1}$	0.5	0.7	$\{0.5/A, 0.5/B\}$
$\mathbf{x^2}$	0.3	0.1	$\{0.9/A, 0.1/B\}$

The values of these weights indicate that $\mathbf{x^1}$ is closer to $\mathbf{z}$ than $\mathbf{x^2}$, but has a greater imperfection. Moreover, the classes of $\mathbf{x^1}$ and $\mathbf{x^2}$ are imprecise.

Using this information and applying the different functions $AggreN()$ and $AggreF()$, we obtain the results shown in Table 2.

Table 2 Results for the different functions $AggreN()$ and $AggreF()$

			AggreN()					
			SVEN		**WSVEN**		**WCVEN**	
		Class	A	B	A	B	A	B
$\mathbf{x^1}$			1	1	0.35	0.35	0.175	0.175
$\mathbf{x^2}$			1	0	0.03	0	0.027	0.003
$\mathbf{z}$	**AggreF**()	**SV**	1	0	1	0	1	0
		CV	0.67	0.33	0.52	0.48	0.53	0.47

Therefore, the various defined aggregation methods obtain the following possible assignments for the class attribute of $\mathbf{z}$:

$\mathbf{SM_{SV}}$ obtains the information $\{A\}$	$\mathbf{SM_{CV}}$ obtains the information $\{0.67/A, 0.33/B\}$
$\mathbf{WSM_{SV}}$ obtains the information $\{A\}$	$\mathbf{WSM_{CV}}$ obtains the information $\{0.52/A, 0.48/B\}$
$\mathbf{WM_{SV}}$ obtains the information $\{A\}$	$\mathbf{WM_{CV}}$ obtains the information $\{0.53/A, 0.47/B\}$

Once the neighbors information is aggregated, the output class inferred by kNN$_{imp}$ classifier will depend on the control parameter U_D. If we define $U_D = 0.05$, the inferred class using any of the aggregation methods will be $\{A\}$. If we define $U_D = 0.1$ the inferred class using the aggregation method WSM$_{\text{CV}}$ will be $\{A, B\}$ and using the other aggregation methods will be $\{A\}$.

4 Experimental framework

Once presented and described the kNN$_{imp}$ classifier, we perform a set of experiments in order to show its behavior and to compare with different techniques of literature. For these purposes, several synthetic datasets with imperfect values are built using NIP tool, [8], from datasets in UCI repository [36]. Also, real datasets with imperfect values are used.

To perform these experiments, the experimental framework is presented in this section. The elements included in the framework are below described.

4.1 Datasets

We use 35 supervised classification datasets taken from the KEEL-dataset [3] and the UCI [36] repositories. Since we will perform various experiments on these datasets directly or by constructing synthetic datasets with imperfect values from them, we will summarize the main characteristics of the datasets in each experiment. Some datasets include instances with missing values that have not been discarded.

In general, datasets are partitioned following a five or ten folds cross-validation procedure [56].

4.2 Parameter configuration

4.2.1 The parameters k, U_D and U_I

In experiments, for kNN$_{imp}$ classifier, we have chosen a representative set of fixed values for the k parameter, $k \in \{1, 3, 5, 7, 9, \sqrt{|E|}\}$.

The kNN$_{imp}$ classifier is executed for each dataset using $U_I = 1$ and $U_D = 0.1$. With $U_I = 1$ all instances are classified regardless of the quality of the neighbors. With $U_D = 0.1$, the classifier returns the class "c" with the highest membership degree μ_c, and those d whose membership degree differ from the highest membership degree at most 0.1, i.e., $\frac{\mu_c - \mu_d}{\mu_c} \leq 0.1$.

4.2.2 The measure $d_{imp}(\cdot, \cdot)$

The $d_{imp}(\cdot, \cdot)$ measure can be defined using different measures of literature, [16,18,32,54], or custom measures. For all experiments, we use the following measure:

- $d_{imp}(\mathbf{x}, \mathbf{y}) = DDP(\mathbf{x}, \mathbf{y})$ defined using the following functions for numerical and nominal attributes:
  - For Numerical Attributes: the distance of Diamond [16] (denoted by D) is defined as:

$$f_1(x_i, y_i) = D(x_i, y_i) = \frac{\sqrt{\frac{(x_i^a - y_i^a)^2 + (x_i^b - y_i^b)^2 + (x_i^c - y_i^c)^2 + (x_i^d - y_i^d)^2}{4}}}{max_i - min_i}$$

    where x_i and y_i represent the i-th attribute of instances $\mathbf{x}$ and $\mathbf{y}$ whose values are defined by the quadruples $(x_i^a, x_i^b, x_i^c, x_i^d)$ and $(y_i^a, y_i^b, y_i^c, y_i^d)$ respectively (as indicated in Section 3.2). max_i,

min_i are the maximum and minimum values of attribute i in the dataset.

– For Nominal Attributes: the measure of Dubois and Prade [18] (denoted by DP) is defined as:

$$f_2(x_i, y_i) = DP(x_i, y_i) = 1 - \frac{Card(x_i \bigcap y_i)}{Card(x_i \bigcup y_i)}$$

where x_i and y_i represent the i-th attribute of instances $\mathbf{x}$ and $\mathbf{y}$ whose values are defined by fuzzy subsets as indicated in Section 3.2. $Card(\cdot)$ is the cardinality of fuzzy sets.

4.2.3 Imperfection measure

Among possible functions, we use the power of the fuzzy sets, [14], that we will apply to both numeric and nominal values.

$$g(x_i) = \begin{cases} \frac{1}{|\Omega_{x_i}|} \sum_{a \in x_i} \mu^i(a) & \text{if } x_i \text{ is fuzzy, interval or crisp/fuzzy subset value} \\ 0 & \text{if } x_i \text{ is crisp value} \end{cases}$$

In fuzzy and interval values, the sum is the integral in the domain where the value is defined.

4.3 Classification Accuracy

Due to the fact that the proposed classifier can return as output in the classification of an instance $\mathbf{z}$ a multi-valued class, we must use a process to measure the classification accuracy of the set of instances considered as test, E_{test}. The proposed process is an adaptation of the process used [9] and [46] to calculate the accuracy in classification when classes are imprecise. This process is shown in Algorithm 4 "Measuring the Classification Accuracy".

Algorithm 4: Measuring the Classification Accuracy

Input Dataset E_{test}, Class value of $\mathbf{z}$ ($class(\mathbf{z})$), Class value inferred to $\mathbf{z}$ (($class_{kNN_{imp}}(\mathbf{z})$)

Suc, $SucErr$=0;

for all $\mathbf{z}$ in E_{test} **do**

  if $class_{kNN_{imp}}(\mathbf{z}) = class(\mathbf{z})$ **then** $Suc = Suc + 1$

  else

    if $(class_{kNN_{imp}}(\mathbf{z}) \bigcap class(\mathbf{z})) \neq \emptyset$ **then** $SucErr = SucErr + 1$

end for

With this process, an interval $[Acc_{min}, Acc_{max}]$ of classification accuracy is obtained, where $Acc_{min} = \frac{Suc}{|E_{test}|}$ and $Acc_{max} = \frac{Suc+SucErr}{|E_{test}|}$. In the definition of the upper bound of this interval we consider as success those cases where the class value of a test instance is not the same but it is included in the inferred class value. Note that situations in which the two values of interval are equal ($Acc_{min} = Acc_{max}$), will be denoted with a single value Acc_{min}.

4.4 Statistical procedure

Finally, we perform a statistical analysis. We use nonparametric statistical tests as recommended in [22, 23]. We use the Friedman test and the Holm's procedure as post-hoc test. When we use the Friedman test, if the null-hypothesis is rejected then there are differences in the performance of the methods, but we do

not know between which of them. In this case, we need a post hoc test (we use the Holm's procedure) to find statistical differences between the different methods. We use the R package [26] for the application of these statistical tests.

For the third experiment, we will show several boxplots with the results. We use the extended boxplots proposed in the paper [46] to represent the intervals obtained with the measure for the classification accuracy. These boxplots represent both the crisp and imprecise results. The box shows the 75% percentile of the maximum error and the 25% percentile of the minimum error. Moreover, the box shows two marks for the median (interval-valued) and two marks for the mean (interval-valued).

5 Experimental study

In this section, we analyze the behavior of kNN$_{imp}$ classifier, based on the experimental framework already described. Although the main objective of this work is to verify that kNN$_{imp}$ classifier has a good behavior when dealing with datasets with imperfect information, we will perform other different experiments. So, this study is organized into three stages:

- In a first stage (Section 5.1) we analyze the behavior of the kNN$_{imp}$ classifier over datasets with imperfect data. These datasets are synthetic and built using NIP tool [8], from datasets of UCI repository [36]. Thus, these datasets will contain the imperfect values commented in Section 3 in addition to the missing values that contain the original datasets.
- In the second stage (Section 5.2) testing the performance of the kNN$_{imp}$ classifier comparing it with other methods of literature based on kNN methods and other classifiers. The datasets used will be those mentioned in these methods of literature.
- A third stage (Section 5.3) testing the performance of the kNN$_{imp}$ classifier over real datasets which contain imperfect data, and compare the results with those obtained with other methods that treat these data.

5.1 Analyzing the behavior with imperfect data

In these experiments, we use several datasets of the UCI repository [36]. Some of these datasets have missing values in a "natural" way. To analyze the performance of the kNN$_{imp}$ classifier, from these classical datasets, we have induced imperfect values (using NIP tool [8]) building synthetic datasets. The datasets are designed as follows: First, we introduce randomly in each dataset a 3% of interval values plus a 3% of fuzzy values in each numeric attribute; and 3% of crisp subsets plus a 3% of fuzzy subsets in each nominal attribute. Note that missing values in addition to the missing values that contain the original datasets are not introduced. Because of this, in bivalued nominal attributes are not added crisp/fuzzy subsets since a missing value would be added. Secondly, we make the same process adding a 5% of imperfect values. In Table 3, we summarize the properties of the synthetic datasets.

In Table 3, for each dataset (with abbreviation "Abbr"), we show the number of instances (|E|), the number of numerical (Nu) and nominal (No) attributes, the number of classes (I), the percentage of missing values (% MV), the percentage of added imperfect values - intervals, fuzzy, crisp and fuzzy subsets - (% IFCFSV), and the percentage of instances with some imperfect value including missing values (%I. with ImV). The last two values are specified for both datasets with 3% and with 5% of imperfect values.

As we can see in Table 3, there are three versions of each dataset: the original dataset, dataset with 3% of added imperfect values and dataset with 5% of added imperfect values. To denote these two latest versions we will use "3%" and "5%". When we make a n folds cross-validation experiment, both the training and the testing partitions will contain imperfect values.

In these experiments, we have made a 10 folds cross-validation. Table 4 shows the accuracy rates obtained, expressed in percentage, the k value and the combination methods that obtain such results. Furthermore, to some datasets, different combination methods obtain good alternative results. So, in the columns with shaded headers and on these datasets, we show other alternative good results expressed as intervals.

Table 3 Description of datasets with imperfection

							3%		5%	
Datasets	**Abbr**	**|E|**	**Nu**	**No**	**I**	**% MV**	**% IFCFSV**	**%I. with ImV**	**% IFCFSV**	**%I. with ImV**
Australian	**AUS**	690	6	8	2		4.35%	46.81%	7.04%	64.35%
Breast Cancer W.	**BCW**	699	9	0	2	0.25%	6.01%	42.63%	10.01%	61.37%
Credit Screen	**CRX**	690	6	9	2	0.65%	4.46%	52.03%	7.23%	69.42%
Glass	**GLA**	214	9	0	6		5.61%	40.65%	10.28%	61.68%
Hepatitis	**HEP**	155	6	13	2	5.64%	2.04%	67.74%	3.26%	76.77%
Horse-colic	**HOR**	368	7	15	2	23.80%	5.16%	100.0%	8.89%	100.0%
Ionosphere	**ION**	351	34	0	2		6.08%	88.03%	9.95%	96.30%
Iris	**IRI**	150	4	0	3		6.67%	25.33%	9.33%	34.67%
Livers disorders	**LIV**	345	6	0	2		5.80%	30.43%	9.86%	46.67%
Monks 1	**MO1**	432	6	0	2		6.02%	30.32%	10.19%	47.69%
Monks 2	**MO2**	432	6	0	2		6.02%	31.71%	10.19%	49.31%
Monks 3	**MO3**	432	6	0	2		6.02%	31.02%	10.19%	48.15%
Mushroom	**MUS**	8124	0	22	2		4.37%	62.59%	7.27%	81.43%
Pima diabetes	**PIM**	768	8	0	2		5.99%	39.45%	9.90%	55.99%
Segmentation	**SEG**	210	19	0	7		5.11%	64.76%	8.52%	83.33%
Soybean-large	**SBL**	307	0	35	19	6.63%	3.02%	72.96%	5.03%	88.60%
Soybean-small	**SBS**	47	0	35	4		0.97%	29.79%	1.95%	53.19%
Wine	**WIN**	178	13	0	3		5.62%	52.25%	10.11%	74.72%
Zoo	**ZOO**	101	1	16	7		0.70%	10.89%	1.16%	18.81%

As we can see in Table 4, the results obtained in 3% and 5% datasets remain very stable with respect to the results of the original datasets. In general, the combination methods that obtain the best results in both synthetic and original datasets are WSM_{SV} and WM_{SV}. Also combination methods WSM_{CV} and WM_{CV} obtain good alternative solutions. With regard to the value of k we can not highlight a particular value to get the best solutions, since this value depends on the size of the training sets for the classifier.

In experiments of Second and Third stages, we will use the combination methods WSM_{SV} and WM_{SV}, and to alternative solutions, the combination methods WSM_{CV} and WM_{CV}.

5.2 Comparison of kNN_{imp} classifier with others classifiers

This section summarizes two experiments performed to observe the effectiveness of the kNN_{imp} classifier when compared with others classifiers:

i) In the first one, we compare with the results presented in [60] and the method proposed in [10]. Among them, we use the kNN algorithm (with normalized Euclidean distance for linear attributes and the overlap metric for nominal attributes) and methods with the best results in [60], i.e., a) IDIBL algorithm (based on k nearest neighbors where the voting weight of each neighbor depends on its distance to the input vector – this weight decreases as the distance grows depends on which kernel function is used), b) BP (the Backpropagation neural network [55]), c) IB1-4 algorithm (four instance-based learning algorithms [1,2]), d) Bayes classifier (a naive Bayesian classifier [40]), and e) DGC+ (gravitation-based classification algorithm, that uses the CMA-ES algorithm to calculate the weighs to describe the importance of each attribute, [10]).

ii) In the second one, we compare with the results presented in [29] and the method proposed in [10]. We use the results obtained with: a) the Weighted Distance Nearest Neighbor method (WDNN method) proposed by authors in [29], b) a modified version of the WDNN method (MWDNN method) [29], c) a Basic-NN, d) a heuristic method based on an adaptive distance measure (A-NN method) proposed in [59], e) a method with a weighted distance based on an algorithm to automatically learn the corresponding weights (PW method) [53], f) an instances reduction method which simultaneously trains both a reduced set of instances and a suitable local metric for these instances (LPD method) [52], and g) DGC+ (gravitation-based classification algorithm, that uses the CMA-ES algorithm to calculate the weighs to describe the importance of each attribute, [10]).

Table 4 kNN$_{imp}$ classifier results for the original datasets and datasets with 3% and 5% of imperfection

Dataset	Introducing imperfect values															
	Without						with 3%						with 5%			
	acc.	k	combination method	acc.	k	combination method	acc.	k	combination method	acc.	k	combination method	acc.	k	combination method	acc.	k	combination method
AUS	86.96	9	SM_{SV},SM_{CV} WSM_{SV},WM_{SV}	[86.96,87.10]	9	WSM_{CV},WM_{CV}	87.39	5	all				87.39	5	all			
BCW	97.42	5	all				97.42	5	all				97.42	5	all			
CRX	87.10	5	all				86.96	7	all				87.10	9	SM_{SV},SM_{CV}			
GLA	71.11	3	WSM_{SV},WM_{SV}	[68.32,75.29]	3	SM_{SV},SM_{CV} WSM_{CV},WM_{CV}	70.64	3	WSM_{SV},WM_{SV}	[67.84,74.82]	3	SM_{SV},SM_{CV} WSM_{CV},WM_{CV}	70.64	3	WSM_{SV},WM_{SV}	[67.84,74.82]	3	SM_{SV},SM_{CV} WSM_{CV},WM_{CV}
HEP	84.84	9	all	[82.41,88.60]	9	WSM_{CV},WM_{CV}	84.84	9	all	[82.41,88.60]	9	WSM_{CV},WM_{CV}	84.84	9	all	[82.41,88.60]	9	WSM_{CV},WM_{CV}
HOR	83.43	7	all				83.68	9	SM_{SV},SM_{CV}				83.69	9	SM_{SV},SM_{CV} WSM_{SV},WM_{SV}			
ION	86.88	1	all				86.88	1	all				87.17	1	all			
IRI	96.00	5	all				96.00	5	all				96.00	5	all			
LIV	67.59	19	WSM_{SV},WM_{SV}	[59.54,72.25]	19	SM_{SV},SM_{CV} WSM_{CV},WM_{CV}	67.88	19	WSM_{SV},WM_{SV}	[59.82,72.55]	19	WSM_{CV},WM_{CV}	67.88	19	WSM_{SV},WM_{SV}			
MO1	86.59	3	all				88.68	3	all				88.45	3	all			
MO2	81.23	5	all				82.85	5	all				82.39	5	all			
MO3	96.75	3	all				96.07	5	all				95.61	5	all			
MUS	100.0	1	all				100.0	1	all				100.0	1	all			
PIM	75.38	7	all				75.25	7	all				75.38	7	all			
SEG	86.67	3	all				86,67	1	all				86.67	1	all			
SBL	91.87	1	all				91.21	1	all				89.92	3	WSM_{SV},WM_{SV}			
SBS	100.0	1	all				100.0	1	all				100.0	1	all			
WIN	96.19	13	all				96.19	13	all	[95.67,97.24]	13	SM_{CV},WSM_{SV} WM_{SV}	96.19	13	all			
ZOO	97.00	1	all				97.00	1	all				96.00	1	all			

i) In this experiment we have used datasets of Table 5. These datasets are selected from [60] and are those with which the methods discussed above gets better results. Table 5 shows |E|, Nu, No, I and % MV as Table 3, and, in addition, shows the percentage of instances with missing (% I. with MV).

Table 5 Datasets description

Datasets	Abbr	|E|	Nu	No	I	% MV	%I. with MV
Australian	**AUS**	690	6	8	2		
Breast Cancer W.	**BCW**	699	9	0	2	0.25%	2.29%
Bridges	**BRI**	105	2	9	6	5.26%	33.33%
Credit Screen	**CRX**	690	6	9	2	0.65%	5.36%
Flag	**FLA**	194	2	26	8		
Glass	**GLA**	214	9	0	6		
Heart disease	**HEA**	270	5	8	2		
Hepatitis	**HEP**	155	6	13	2	5.64%	48.30%
Horse-colic	**HOR**	368	7	15	2	23.80%	98.10%
Ionosphere	**ION**	351	35	0	2		
LED	**LED**	500	7	0	10		
Livers disorders	**LIV**	345	6	0	2		
Pima diabetes	**PIM**	768	8	0	2		
Promoters	**PRO**	106	0	57	2		
Sonar	**SON**	208	60	0	2		
Zoo	**ZOO**	101	1	16	7		

As in [60], we have made a 10 folds cross-validation. Table 6 shows the accuracy rates obtained, expressed in percentage, and the value of k used. As we discussed in the classifier description, some results are expressed as an interval because that interval is the best result or a good alternative solution.

Table 6 Accuracy rates of the kNN$_{imp}$ classifier in comparison with other methods using datasets of the Table 5

Datasets	**kNN** [60]	**IDIBL** [60]	**BP** [60]	**IB1-4** [60]	**Bayes** [60]	**DGC+** [10]	**kNN$_{imp}$ classifier**	
AUS	81.16	85.36	84.50	81.00	83.10	83.74	86.96 (9)	[86.96,87.10] (9)
BCW	95.28	97.00	96.30	96.30	93.60	96.28	97.42 (5)	
BRI	53.73	63.18	67.60	60.60	66.10	62.86	64.73 (3)	
CRX	81.01	85.35	85.10	81.30	82.20	84.93	87.10 (5)	
FLA	48.84	57.66	58.20	56.60	52.50	65.98	60.37 (9)	
GLA	70.52	70.56	68.70	71.10	71.80	70.36	71.11 (3)	[68.32,75.29] (3)
HEA	75.56	83.34	82.60	79.60	75.60	84.52	81.85 (7)	
HEP	77.50	81.88	68.50	79.60	57.50	86.28	84.84 (9)	
HOR	60.82	73.80	66.90	64.80	68.60	85.60	83.43 (7)	
ION	86.33	87.76	92.00	86.30	85.50	93.11	86.88 (1)	
LED	57.20	74.88	69.00	68.50	68.50	48.80	[71.80,72.80] (22)	[68.80,77.60] (22)
LIV	63.47	62.93	69.00	62.30	64.60	64.93	67.59 (19)	
PIM	70.31	75.79	75.80	70.40	72.20	74.51	75.38 (7)	
PRO	82.09	88.64	87.90	82.10	78.20	90.56	79.24 (10)	
SON	86.60	84.12	76.40	86.50	73.10	84.87	83.11 (1)	
ZOO	94.44	92.22	95.60	96.70	97.80	95.53	97.00 (1)	
Average	74.05	79.03	77.76	76.48	74.43	79.55	[79.93,79.99]	[79.56,80.56]

For these datasets, the results obtained clearly show that the proposed kNN$_{imp}$ classifier obtains a good accuracy. We make a statistical analysis of the results of best accuracy, where for the result expressed as an interval value we take the lower end (pessimistic value). When the statistic test on these results is performed, firstly the Friedman test is applied, obtaining a rejection of the null-hypothesis (p-value=9.87e-05) with a $\alpha = 0.01$. That is, it rejects that there are no significant differences. When we perform the post-hoc test to the hypotheses of comparison between the methods, the obtained p-values are 0.00252, 0.15132, 0.15132, 0.00336, 0.00174 and 0.58960. Holm's procedure rejects the null-hypotheses to kNN, IB1-4 and Bayes methods, indicating that kNN$_{imp}$ classifier is statistically better regarding accuracy

than these methods (α=0.05). With regard to IDIBL, BP and DGC+ methods there are no significant differences.

ii) In this experiment we have used datasets in [29] (Table 7). These datasets do not have missing values. In Table 7 we show the same characteristics listed above.

Table 7 Datasets description

Datasets	Abbr	|E|	Nu	No	I
Australian	**AUS**	690	34	8	2
Balance	**BAL**	625	4	0	3
Breast Cancer W.	**BCWs**	683	9	0	2
DNA	**DNA**	3186	180	0	3
German	**GER**	1000	11	13	2
Glass	**GLA**	214	9	0	6
Heart disease	**HEA**	270	6	7	2
Liver disorders	**LIV**	347	6	0	2
Pima diabetes	**PIM**	768	8	0	2
Satimage	**SAT**	6435	36	0	6
Vehicle	**VEH**	846	18	0	4
Vote	**VOT**	435	0	16	2
Wine	**WIN**	178	13	0	3

As in [29], we have made a 5 folds cross-validation. Table 8 shows the results obtained indicating the accuracy rates expressed in percentage (value in brackets indicates the value of k used). As we discussed in the classifier description, some results of kNN$_{imp}$ classifier are expressed as an interval because that interval is the best result or a good alternative solution.

Table 8 Accuracy rates of the proposed classifier in comparison with other methods using datasets of the Table 7

Datasets	WDNN [29]	MWDNN [29]	Basic-NN [29]	A-NN [59]	PW [53]	LPD [52]	DGC+	kNN$_{imp}$ classifier	
AUS	85.48	85.01	81.36	75.91	83.50	86.10	84.93	87.10 (7)	[86.23,88.12] (7)
BAL	89.14	90.32	69.29	89.88	86.56	83.70	89.60	89.92 (25)	
BCWs	97.52	97.88	96.76	97.14	96.68	96.60	96.18	97.66 (5)	
DNA	96.15	95.84	87.06	92.73	93.51	95.10	95.17	93.67 (56)	
GER	75.84	73.89	71.13	61.89	71.68	74.00	72.30	75.00 (32)	
GLA	71.34	70.81	68.26	71.22	73.72	72.00	68.22	73.00 (3)	[70.22,76.27] (3)
HEA	83.91	84.91	76.34	67.45	81.06	81.40	82.96	83.33 (16)	
LIV	68.31	65.39	64.78	65.12	63.78	66.70	69.56	66.96 (19)	
PIM	75.96	76.31	70.83	71.86	72.61	74.00	76.04	75.26 (28)	
SAT	89.88	89.01	88.29	90.49	91.20	89.40	87.54	91.32 (5)	
VEH	70.14	69.15	70.43	66.28	70.69	72.60	70.57	72.70 (5)	
VOT	92.29	91.37	92.86	93.31	94.49	96.30	94.94	94.25 (7)	
WIN	96.61	96.04	97.29	84.82	98.65	95.00	96.63	97.76 (13)	
Average	84.04	83.53	79.59	79.08	82.93	83.30	83.43	84.48	[84.20,84.81]

Also, for these datasets, the results obtained clearly show that the kNN$_{imp}$ classifier obtains a good accuracy. Finally, we perform the statistic analysis on these results. We first apply the Friedman test, obtaining a rejection of the null-hypothesis (p-value=0.0003061) with a $\alpha = 0.01$. That is, it rejects that there are no significant differences. When we perform the post-hoc test to the hypotheses of comparison between the kNN$_{imp}$ classifier and the others, we obtain their p-values: 0.13670, 0.05622, 0.00085, 0.00290, 0.01994, 0.05622, 0.07420. Holm's procedure rejects the null-hypotheses to Basic-NN, A-NN and PW methods indicating that the kNN$_{imp}$ classifier is statistically better regarding accuracy than these methods (α=0.05). With a α=0.06, kNN$_{imp}$ classifier is statistically better regarding accuracy than MWDNN and LPD methods. And with regards to WDNN and DGC+ methods, there are no significant differences.

5.3 Comparing with imperfect real datasets

The aim of these experiments is to apply the kNN$_{imp}$ classifier to imperfect real datasets (available on "http://sci2s.ugr.es/keel/"). These datasets have attributes described by interval values and multi-valued classes (a more detailed description may be found in [46]). We compare the results obtained by the kNN$_{imp}$ classifier with the ones obtained by the FRF ensemble [7,9] and the GFS classifier [46].

In Table 9, "% IV" denotes the % of interval values and "%I with ImV" denotes the % of the instances with some imperfect value.

Table 9 Datasets description

Datasets	**|E|**	**Nu**	**No**	**I**	**% IV**	**% I with imV**
Long-4	25	4	0	2	100	100
100ml-4-I	52	4	0	2	100	100
100ml-4-P	52	4	0	2	100	100

As in [9,46], we have used a 10 folds cross-validation for all datasets. As we said before, we use the combination methods WSM$_{SV}$ and WM$_{SV}$.

In Table 10 the results obtained in [9,46] and the kNN$_{imp}$ classifier (with $k = 1$) are shown. The results obtained with the kNN$_{imp}$ classifier with the two combination methods show that the proposed classifier has a good performance when compared with the other methods.

Table 10 Error rates with imperfect real datasets

Technique	100ml-4-I	100ml-4-P	Long-4
Crisp [46]	0.38	0.42	0.54
GGFS [46]	[0.19,0.48]	[0.17,0.41]	[0.35,0.62]
FRFMIWLF1 [9]	[0.27,0.47]	[0.10,0.30]	[0.18,0.45]
FRFSM2 [9]	[0.23,0.43]	[0.10,0.30]	[0.22,0.48]
kNN$_{imp}$ classifier	[0.0,0.13]	[0.04,0.16]	[0.23,0.50]

As we have commented in Section 4.4, in Figure 1, the extended boxplots with the results are shown. The boxplots show the results expressing the error. The dotted lines show the means (interval-valued) and the continuous lines show the respective medians (interval-valued).

The results obtained by kNN$_{imp}$ classifier are very promising because we are representing the information in an appropriate and more natural way, and the accuracy rates are improved.

6 Conclusions

In this work the neighborhood based method, kNN$_{imp}$ classifier, has been presented. This method allows us to carry out the classification from datasets with different types of imperfect values simultaneously. Thus, the proposed classifier does not need previous data transformations to perform the classification task respecting the real nature of them. In this method, the importance of each neighbor is based on relative distance and its degree of imperfection. The method uses external parameters to limit the allowed imperfection and how close to the best one must be a class to be included in the classification of an instance. As future work, a detailed analysis of the influence of measures and parameters should be performed. In addition, we will carry out the extension of the kNN$_{imp}$ classifier for imputing missing values of any attribute from datasets with imperfect values.

The results, validated through statistical tests, show that the kNN$_{imp}$ classifier is robust when working with imperfect data. kNN$_{imp}$ classifier maintains a good performance when compared with other methods in the literature, applied to datasets without imperfection. In the same way it has a good performance when compared with other methods, applied to datasets with imperfect values.

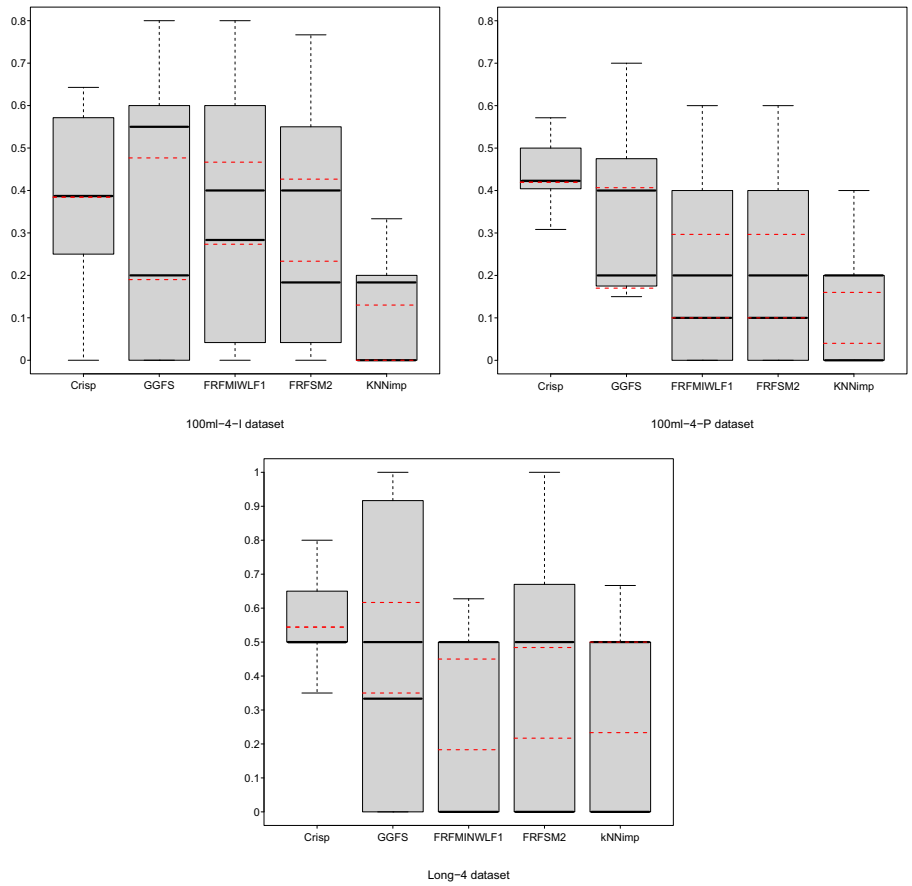

Fig. 1 Extended boxplots for the results of the Table 10

Acknowledgement

Supported by the project TIN2014-52099-R (EDISON) granted by the Ministry of Economy and Competitiveness of Spain (including ERDF support).

Compliance with Ethical Standards

Funding

This study was funded by the research project of Ministry of Economy and Competitiveness TIN2014-52099-R (EDISON), including European Regional Development Fund.

Conflict of Interest

The authors declare that they have no conflict of interest.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Aha DW (1992) Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. International Journal of Man-Machine Studies 36(2): 267–287
2. Aha DW, Kibler D, Albert KM (1991) Instance-based learning algorithms. Machine Learning 6(1): 37–66
3. Alcalá-Fdez J, Fernández A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011) Keel data-mining software tool: data set repository, integration of algorithm and experimental analysis framework. Journal of Multiple-Valued Logic and Soft Computing 17(2-3): 255–287
4. Barua A, Mudunuri LS, and Kosheleva O (2014) Why Trapezoidal and Triangular Membership Functions Work So Well: Towards a Theoretical Explanation. Journal of Uncertain System 8(3): 164–168
5. Berlanga F, Rivas AR, del Jesús M, Herrera F (2010) Gp-coach genetic programming-based learning of compact and accurate fuzzy rule-based classification systems for high-dimensional problems. Information Science 180(8): 1183–1200
6. Bezdek J (1981) Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum, New York, USA
7. Bonissone PP, Cadenas JM, Garrido MC, Díaz-Valladares RA (2010) A fuzzy random forest. International Journal of Approximate Reasoning 51(7): 729–747
8. Cadenas JM, Garrido MC, Martínez R (2013) Nip – an imperfection processor to data mining datasets. International Journal of Computational Intelligence Systems 6(1): 3–17
9. Cadenas JM, Garrido MC, Martínez R, Bonissone PP (2012) Extending information processing in a fuzzy random forest. Soft Computing 16(6): 845–861
10. Cano A, Zafra A, Ventura S (2013) Weighted Data Gravitation Classification for Standard and Imbalanced Data. IEEE Transactions on Cybernetics 43(6): 1672–1687
11. Clare A, King R (2001) Knowledge discovery in multi-label phenotype data. In: Proceedings of the 5th European Conference on Principles of data mining and knowledge discovery, Freiburg, Germany, pp 42-53
12. Cover T, Hart PE (1967) Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1): 21–27
13. Crockett K, Bandar Z, Mclean D (2001) Growing a fuzzy decision forest. In: Proceedings of the 10th IEEE International Conference on Fuzzy Systems, Melbourne, Victoria, Australia, pp 614-617
14. DeLuca A, Termini S (1972) A definition of a nonprobabilistic entropy in the setting of fuzzy sets theory. Information and Control 20(4): 301–312
15. Derrac J, García S, Herrera F (2014) Fuzzy nearest neighbor algorithms: Taxonomy, experimental analysis and prospects. Information Sciences 260: 98–119
16. Diamon P, Kloeden P (1994) Metric spaces of fuzzy sets: Theory and application. World Scientific Publishing, London, UK
17. Dombi J, Porkolab L (1991) Measures of fuzziness. Annales Universitasis Scientiarium Budapestinensis, Sectio Computatorica 12: 69–78
18. Dubois D, Parde H (1980) Fuzzy sets and system: Theory and applications. Academic Press, New York, USA
19. Duda RO, Hart PE, Stork DG (2001) Pattern classification. John Wiley and Sons, New York, USA
20. Fernández A, del Jesús M, Herrera F (2009) Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets. International Journal of Approximate Reasoning 50(3): 561577
21. Fix E, Hodges J (1989) Discriminatory analysis, nonparametric discrimination: Consistency properties. International Statistical Review 57(3): 238–247
22. García S, Fernández A, Luengo J, Herrera F (2009) A study statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Computing 13(10): 959–977
23. García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. Information Sciences 180(10): 2044–2064
24. Garrido MC, Cadenas JM, Bonissone PP (2010) A classification and regression technique to handle heterogeneous and imperfect information. Soft Computing 14(11): 1165–1185
25. Huang Z (2002) A fuzzy k-modes algorithm for clustering categorical data. IEEE Transactions on Fuzzy Systems 7(4): 446–452
26. Ihaka R, Gentleman R (1996) R: A language for data analysis and graphics (http://www.r-project.org/). Journal of Computational and Graphical Statistic 5(3): 299–314
27. Inoue T, Abe S (2001) Fuzzy support vector machines for pattern classification. In: Proceedings of International Joint Conference on Neural Networks, Washington, DC, USA, pp 1449–1454
28. Ishibuchi H, Yamamoto T (2005) Rule weight specification in fuzzy rule-based classification systems. IEEE Transactions on Fuzzy Systems 13(4): 428436
29. Jahromi MZ, Parvinnia E, John R (2009) A method of learning weighted similarity function to improve the performance of nearest neighbor. Information Sciences 179(17): 2964–2973
30. Janikow CZ (1998) Fuzzy decision trees: issues and methods. IEEE Transaction on Systems, Man, and Cybernetics, Part B 28(1): 1–14
31. Janikow CZ (2003) Fuzzy decision forest. In: Proceedings of the 22nd International Conference of the North American Fuzzy Information Processing Society, Chicago, USA, pp 480–483
32. Johanyák ZC, Kovács S (2005) Distance based similarity measures of fuzzy sets. In: Proceedings of the 3rd Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence, Herlany, Slovakia, pp 265–276
33. Kaufmann A (1975) Introduction to the theory of fuzzy subsets: Fundamental theoretical elements. Academic Press, New York, USA
34. Lee K, Lee K, Lee J (1999) A fuzzy decision tree induction method for fuzzy data. In: Proceedings of IEEE International Fuzzy Systems Conference, Seoul, South Korea, pp 16–21

35. Li D, Gu H, Zhang L (2010) A fuzzy c-means clustering algorithm based on nearest-neighbor intervals for incomplete data. Expert Systems with Applications 37(10): 6942–6947
36. Lichman M (2013) UCI Machine Learning Repository, http://archive.ics.uci.edu/ml, University of California, School of Information and Computer Sciences, Irvine, CA
37. Lin C, Wang S (2002) Fuzzy support vector machines. IEEE Transactions on Neural Network 13(2): 464471
38. Madjarov G, Kocev D, Gjorgjevikj D, Džeroski S (2012) An extensive experimental comparison of methods for multi-label learning. Pattern Recognition 45(9): 30843104
39. Marsala C (2009) Data mining with ensembles of fuzzy decision trees. In: Proceedings of IEEE Symposium on Computational Intelligence and Data Mining, Nashville, TN, USA, pp 348–354
40. Michie D, Spiegelhalter D, Taylor C (1994) Machine learning, neural and statistical classification. Ellis Horwood, Upper Saddle River, NJ, USA
41. Mitra S, Pal SK (1995) Fuzzy multi-layer perceptron, inferencing and rule generation. IEEE Transactions on Neural Networks 6(1): 51–63
42. Moore RE (1979) Methods and applications of interval analysis. (SIAM) Studies in Applied Mathematics 2, Soc for Industrial & Applied Math, Philadelphia
43. Nauck D, Krusel R (1997) A neuro-fuzzy method to learn fuzzy classification rules from data. Fuzzy Sets and Systems 89(3): 277–288
44. Olaru C, Wehenkel L (2003) A complete fuzzy decision tree technique. Fuzzy Sets and Systems 138(2): 221–254
45. Otero A, Otero J, Sánchez L, Villar JR (2006) Longest path estimation from inherently fuzzy data acquired with GPS using genetic algorithms. In: Proceedings of the International Symposium on Evolving Fuzzy Systems, Lancaster, UK, pp 300–305
46. Palacios AM, Sánchez L, Couso I (2009) Extending a simple genetic cooperative-competitive learning fuzzy classifier to low quality datasets. Evolutionary Intelligence 2(1): 73–84
47. Palacios AM, Sánchez L, Couso I (2010) Diagnosis of dyslexia with low quality data with genetic fuzzy systems. International Journal of Approximate Reasoning 51(8): 993–1009
48. Palacios AM, Sánchez L, Couso I (2011) Future performance modeling in athletism with low quality data-based genetic fuzzy systems. Journal of Multiple-Valued Logic and Soft Computing 17: 207–228
49. Palacios AM, Sánchez L, Couso I (2012) Boosting of fuzzy rules with low quality data. Journal of Multiple-Valued Logic and Soft Computing 19: 591–619
50. Palacios AM, Sánchez L, Couso I (2013) An extension of the furia classification algorithm to low quality data. In: Hybrid Artificial Intelligent Systems (LNCS 8073), JS Pan, MM Polycarpou, M Wozniak, ACPLF de Carvalho, H Quintián, E Corchado (eds.) Springer-Verlag, Berlin, pp 679–688
51. Palacios AM, Palacios JL, Sánchez L, Alcalá-Fdez J (2015) Genetic learning of the membership functions for mining fuzzy association rules from low quality data. Information Sciences 295, 358–378
52. Paredes R, Vidal E (2006) Learning prototypes and distances: a prototype reduction technique based on nearest neighbor error minimization. Pattern Recognition 39(2): 180–188
53. Paredes R, Vidal E (2006) Learning weighted metrics to minimize nearest-neighbor classification error. IEEE Transaction on Pattern Analysis and Machine Intelligence 28(7): 1100–1110
54. Ralescu AL, Ralescu DA (1984) Probability and fuzziness. Information Science 34(2): 85–92
55. Rumelhart DE, Mcclelland JL (1986) Parallel distributed processing. MIT Press, Cambridge, MA, USA
56. Stone M (1974) Cross-validatory choice and assessment of statistical predictions. Journal of the Royal Statistical Society, Series B (Methodological) 36(2): 111–147
57. Torra V (2005) Fuzzy c-means for fuzzy hierarchical clustering. In: Proceedings of the 14th IEEE International Conference on Fuzzy Systems, Reno, NV, USA, pp 646–651
58. Villar J, Otero A, Otero J, Sánchez L (2009) Taximeter verification using imprecise data from GPS. Engineering Applications of Artificial Intelligence 22(2): 250–260
59. Wang J, Neskovic P, Cooper LN (2007) Improving nearest neighbor rule with a simple adaptive distance measure. Pattern Recognition Letters 28(2): 207–213
60. Wilson DR, Martinez TR (2000) An integrated instance-based learning algorithm. Computational Intelligence 16(1): 1–28
61. Witten IH, Frank E, Hall MA (2011) Data mining (Third edition). Morgan Kaufmann Publishers, San Francisco, CA, USA
62. Younes Z, Abdallah F, Denoeux T (2010) Fuzzy multi-label learning under veristic variables. In: Proceedings of the IEEE International Conference on Fuzzy Systems, Yantai, China, pp 1–8
63. Zadeh L (1965) Fuzzy sets. Information and Control 8: 183–190