eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Unibertsitate Masterra

Konputazio Ingeniaritza eta Sistema Adimentsuak

Konputazio Zientziak eta Adimen Artifiziala Saila –
– Departamento de Ciencias de la Computación e Inteligencia Artificial

K I S A

I C S I

Master Tesia

Testing Modified Confusion Entropy as split criterion for decision trees.

Alexander Gonzalo de Sá

Tutorea(k)

David Núñez González
Department of Computer Science and Artificial Intelligence
Faculty of Informatics

Master eta Doktorego Eskola
Escuela de Máster y Doctorado
Master and Doctoral School

KZAA
/CCIA

2019ko maiatzan

Master eta Doktorego Eskola
Escuela de Máster y Doctorado
Master and Doctoral School

Abstract

In 2010, a new performance measure to evaluate the results obtained by algorithms of data classification was presented, *Confusion Entropy (CEN)*. This render measure is able to achieve a greater discrimination than *Accuracy* focusing on the distribution across different classes of both correctly and wrongly classified instances, but it is not able to work correctly in cases of binary classification. Recently, an enhancement has been proposed to correct its behaviour in those cases, the *Modified Confusion Entropy (MCEN)*.

In this work, we propose a new algorithm, *MCENTree*. This algorithm uses *MCEN* as splitting criterion to build a decision tree model instead of *CEN*, as proposed in the *CENTree* algorithm in the literature.

We make a comparison among a classic J48, *CENTree* and the new algorithm *MCENTree* in terms of Accuracy, *CEN* and *MCEN* performance measures, and we analyze how the undesired behaviour of *CEN* affects the results of the algorithms and how *MCEN* shows a good behaviour in terms of results: while *MCENTree* gives correct results in a statistical range [0,1], *CENTree* sometimes gives non monotonous and out of range results in binary class classification.

Contents

Chapter 1

Introduction

1.1 Context

Data mining science let us extract knowledge from huge amount of data. There are different steps along the data mining lifecycle that have been covered to reach the extracted knowledge. These steps include Business/Research Understanding Phase, Data Understanding Phase, Data Preparation Phase, Modeling Phase, Evaluation Phase, and Deployment Phase[1]:

1. Business/Research Understanding Phase: In this phase the objectives of the project are listed, translated to the definition of the data mining problem and a strategy to fulfill these objectives is planned.

2. Data Understanding Phase: In this phase a preliminary analysis of the data is done to familiarize with it and check its quality. Target data can also be discerned in this phase.

3. Data Preparation Phase: The objective of this phase is to prepare the raw data for the modeling algorithms. For this, some cases can be selected and some variables can be selected or transformed. Inconsistent raw data must be cleaned or transformed into an understandable format for the modeling algorithm. Problems such as unbalanced data or missing values must be faced too.

4. Modeling Phase: Sometimes different modeling techniques can be used for the data, and each of those techniques have their own different settings. The goal is find the model and set the configuration that

optimizes the classification results. For this purpose it could also be necessary to go back to the data preparation phase and check the results obtained with different configurations of the data.

5. Evaluation Phase: An analysis of the results obtained with the different models has to be done to check which one gets better results and if it accomplishes the objectives of the project.

6. Deployment Phase: Make use of the created model in a real case scenario, with real data. Write a report with the obtained results.

This work focuses on the evaluation phase, where an evaluation of the performance of machine learning models is made. With that purpose, different metrics are classically used, like *Accuracy*, *Precision*, *Area Under ROC Curve* score, Matthews correlation coefficient (*MCC*)[2] , or more recently, Confusion Entropy (*CEN*), a metric based in Shannon's Entropy [3] proposed by Wei et al. [4] in 2010.

This new metric, published as a more discriminant metric than *Accuracy*, not only has the percentage of well classified objects in consideration like *Accuracy* does, but also considers the distribution of wrong classified objects across the classes. Thus, *CEN* gives an overview on how classification have been done in terms of number of elements in the main diagonal or outside the main diagonal. In other words, when the main diagonal of the confusion matrix contains the maximum number of possible elements it means that there has been a good classification and therefore the entropy (or disorder) is minimal. Conversely, when the maximum number of possible elements are outside the main diagonal it means that there has been a bad classification and therefore the entropy (or disorder) is maximum.

But *CEN* [5], presents an anomaly in the calculation of entropy for binary classification. Since it is supposed to be a statistic in [0,1] range it presents a non-monotonous behaviour with results greater than 1. An improvement is given by [6][7] where this anomaly is corrected.

1.2 Motivation

Since many works have been based on *CEN* for different purposes and a new definition of *CEN* have been given (known as *MCEN*)it is interesting to review these works to check how the new definition affects. Thus, the goal of this work is to compare the results obtained by the original version of

CEN with the results obtained by the mentioned enhanced version, *MCEN*, in a specific case of classification made with a decision tree algorithm (according to [8]). Consequently, this work aims to help to determine the true worthiness of the *MCEN* in the evaluation of performance of decision trees classification algorithms.

For this purpose, different two-class datasets as those are the ones in which the first version of *CEN* shows its wrong behaviour, are classified through an unique version of the C4.5 algorithm, called *CENTree*, a decision tree constructed based on the Confusion Entropy value.

Then, the results of these classifications are measured with both versions of *CEN* and two classic measure, *Overall Accuracy* and *Matthews Correlation Coefficient*.

1.3 Structure of the document

This document is structured as follows:

- Chapter 1: Introduction. A brief introduction of the work's subject matter and its context.
- Chapter 2: Related works on Confusion Entropy. An introduction to the Confusion Entropy and a study of the related work done to date.
- Chapter 3: The proposed algorithm, MCENTree. In this chapter the MCENTree algorithm is introduced.
- Chapter 4: Material and methods. A description of the data-sets and methods used in the work.
- Chapter 5: Results. Exhibition of the results obtained with the used methods.
- Chapter 6: Conclusions and future work. Summary of the results of the work and new research work lines for the future.

Chapter 2

Related works on Confusion Entropy

In this chapter the work done up until now on the fields involved in this work is studied. In section 2.1 Confusion Entropy and its uses on the bibliography are explained (section 2.1.1). And in section 2.2 the new version of Confusion Entropy (*MCEN*) is described.

2.1 Confusion Entropy

Although being the most used and simpler metric of performance, Accuracy has been widely questioned as the best way to measure results of multi-class classification problems since long ago (Provost, F. J., Fawcett, T., Kohavi, R., 1998 [9]) due to its incapacity to consider the misclassified items. For various classifiers constructed by different methods, the results measured by accuracy can be very similar or even the same, while the misclassified samples could be differently classified. This implies a serious inconvenience in cases where a discrimination between classes is needed.

With the purpose of creating a new metric able to discern the differences between such cases, Confusion Entropy (*CEN*) was introduced in 2010 by Wei et al. [4].

This new metric does not only measure the percentage of well classified items but also the distribution of wrong classified items in a confusion matrix. This way, *CEN* gets better results in cases where wrong classified items have a higher concentration in between different classes than cases where the wrong classified items have a more uniform distribution. These results are shown in a range between *0* and *1*, where *0* is the result of a perfect classification.

	(a)	(b)
	$\begin{pmatrix} 10 & 1 & 1 & 1 \\ 1 & 10 & 1 & 1 \\ 1 & 1 & 10 & 1 \\ 1 & 1 & 1 & 10 \end{pmatrix}$	$\begin{pmatrix} 10 & 3 & 0 & 0 \\ 0 & 10 & 3 & 0 \\ 0 & 0 & 10 & 3 \\ 3 & 0 & 0 & 10 \end{pmatrix}$
ACC=	0.7692	0.7692
CEN=	0.4196	0.2781

Table 2.1: Example of confusion matrices with same *Accuracy* but different *CEN*. Confusion matrix (b) obtains a better *CEN* result for a higher concentration of the misclassified items.

	Baseline	(a)			(b)		
	$\begin{pmatrix} \mathbf{3} & \mathbf{3} \\ \mathbf{3} & \mathbf{3} \end{pmatrix}$	$\begin{pmatrix} 2 & \mathbf{3} \\ \mathbf{3} & 4 \end{pmatrix}$	$\begin{pmatrix} 1 & \mathbf{3} \\ \mathbf{3} & 5 \end{pmatrix}$	$\begin{pmatrix} 0 & \mathbf{3} \\ \mathbf{3} & 6 \end{pmatrix}$	$\begin{pmatrix} \mathbf{3} & 2 \\ 4 & \mathbf{3} \end{pmatrix}$	$\begin{pmatrix} \mathbf{3} & 1 \\ 5 & \mathbf{3} \end{pmatrix}$	$\begin{pmatrix} \mathbf{3} & 0 \\ 6 & \mathbf{3} \end{pmatrix}$
Entropy=	1.0000	0.9183	0.6500	0.0000	0.9183	0.6500	0.0000
ACC=	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000

Table 2.2: Examples for binary case with $S = 12$. (a): Entropy reduction within the main diagonal, IN. (b) Entropy reduction outside the main diagonal, OUT. [7]

Furthermore, *CEN* is compared with Accuracy and other measures in [4], evidencing a relative consistency between them: results with high Accuracy tend to have low *CEN*. Also, authors of [10] show a relation between *CEN* and *MCC*, and how they both have more desirable characteristics than accuracy.

As described in the original publication [4], a multi-class classifier learned from a training data-set, with $N \geq 2$ classes labelled as $\{1, 2, \ldots, N\}$, is used to classify items of a test data-set, or in other words, predict the class of these items with the data of their attributes. Since this work is focused in supervised classification, classes of these test data-set entries are known and a $N \times N$ confusion matrix $C = (C_{i,j})_{i,j=1,\ldots,N}$ can be built, collecting the results of the classification of the test data-set. $C_{i,j}$ is the number of items of class i wrongly classified as class j. S is the notation used to represent the sum of values of the confusion matrix, which corresponds to the total number of entries in the test data-set, $S = \sum_{i=1}^{N} \sum_{j=1}^{N} C_{i,j}$.

The probability for an element of class i to be classified as class j subjected to class j, $P^j_{i,j}$, is shown as:

$$P^j_{i,j} = \frac{C_{i,j}}{\sum_{k=1}^{N} (C_{j,k} + C_{k,j})}, \; i, j = 1, ..., N, \; i \neq j \tag{2.1}$$

Therefore, $P^j_{i,j}$ should be the relative frequency of cases, or cases of class i classified as class j among all cases that are of class j or have been classified as class j. But the results does not reflect that, as the well classified cases of class j, $C_{j,j}$, are counted twice in the denominator [6][7].

In the same way, the probability for an elements of class i to be classified as class j, subjected to class i, $P^i_{i,j}$, is defined as:

$$P^i_{i,j} = \frac{C_{i,j}}{\sum_{k=1}^{N} (C_{i,k} + C_{k,i})}, \; i, j = 1, ..., N, \; i \neq j \tag{2.2}$$

Thus, the value of *CEN* associated to class j is defined in the following way:

$$\text{CEN}_j = - \sum_{k=1, k\neq j}^{N} \left(P^j_{j,k} \log_{2(N-1)}(P^j_{j,k}) + P^j_{k,j} \log_{2(N-1)}(P^j_{k,j}) \right) \tag{2.3}$$

Lastly, *CEN* associated to the confusion matrix is the combination of the *CEN* associated to each class:

$$\text{CEN} = \sum_{j=1}^{N} P_j \, \text{CEN}_j \tag{2.4}$$

Where the non-negative weights P_j, summing 1, are:

$$P_j = \frac{\sum_{k=1}^{N} (C_{j,k} + C_{k,j})}{2 \sum_{k,\ell=1}^{N} C_{k,\ell}} \, . \tag{2.5}$$

2.1.1 Applications

Since Confusion entropy was introduced, aside from the mentioned publications *A unifying view for performance measures in multi-class prediction* [5], where its anomaly was revealed when the authors made a review of the different performance measures in multi-class prediction, and *Enhancing Confusion Entropy as Measure for Evaluating Classifiers*[6], where the improvement was published in 2018 for the first time, it has been used in several cases of study.

Giuseppe Jurman, who previously revealed the error of *CEN* in 2010, published [10] in 2012, where the correlation between *CEN* and *MCC* is analyzed, showing a high correlation between them.

In 2013, *CEN* was used among other metrics to measure different methods performance when classifying highly imbalanced ICU (Intensive Care Unit) data [11].

It was also used that year as comparison for another novel measure, probabilistic accuracy (*Pacc*), in [12].

In 2014, authors of [13] explained and considered using *CEN* to evaluate their proposed novel classifier based on a combination of classic classification methods, but ended up discarding it for its erratic behavior in binary classification cases.

In 2015, in [14] authors used *CEN* as well as Accuracy and *MCC* to evaluate classifications done with data concerning gait analysis with three different assistive devices.

Two different articles where *CEN* is used where published in both 2017 and 2018. In 2017, in [15] Confusion entropy is used in the frame of unsupervised machine learning, in the development of a new collaborative clustering algorithm. It is used as stopping criterion, comparing different clustering algorithms to check if the partitions they work with are identical.

Also, authors of [16] use *CEN* as metric to analyze the probability sensitivity of Gaussian processes in bankruptcy prediction context.

In 2018, *CEN* was used among other metrics in [17] to analyze river water quality prior to its use for drinking according the presence and abundance of some benthic families and the relation with microbial pathogen standards using decision tree models. Some *CEN* results presented in this article surpass, in fact, the theoretical upper limit of *1*.

The authors of [18] used *CEN* to identify patients who are more likely to be readmitted to an intensive care unit.

Both *CEN* and the enhanced version *MCEN* are currently available in the Python library *pycm*, developed to work with confusion matrices of multi-class problems [19].

2.2 Modified Confusion Entropy: *MCEN*

However, in the aforementioned *A unifying view for performance measures in multi-class prediction* [5] (Jurman et al., 2010), while studying the correlation between *CEN* and *MCC*, authors found that *CEN* cannot be reliably used in cases of binary classification. In those cases, when there are only two classes, *CEN* can surpass its theoretical maximum limit of *1*.

In 2018, with the publication of *Enhancing Confusion Entropy as Measure for Evaluating Classifiers*[6], a new version of *CEN* was published. This new version introduced a correction that solves this erratic behavior in cases of binary classification.

In this publication, the formula to calculate the probability of classifying elements of class i as elements of class j subjected to class j is replaced with the following formula:

$$\widetilde{P}_{i,j}^{j} = \frac{C_{i,j}}{\sum_{k=1}^{N}(C_{j,k} + C_{k,j}) - C_{j,j}},\ i,j = 1,...,N,\ i \neq j$$

This way the problem of counting twice the correctly classified cases in the denominator is solved, reflecting in a correct way the relative frequency of cases of class i classified as class j among all cases that are of class j or that have been wrongly classified as class *j*.

The formula for (2.2) is modified in the same way:

$$\widetilde{P}_{i,j}^{i} = \frac{C_{i,j}}{\sum_{k=1}^{N}(C_{i,k} + C_{k,i}) - C_{i,i}},\ ,i,j = 1,...,N,\ i \neq j$$

Making $\widetilde{P}^i_{i,j}$ a correct formula to represent the relative frequency of cases of class i classified as class j among all cases that are of class i or that have been wrongly classified as class i.

The formula for the weights in 2.5 is also modified:

$$\widetilde{P}_j = \frac{\sum_{k=1}^{N} (C_{j,k} + C_{k,j}) - C_{j,j}}{2 \sum_{k,\ell=1}^{N} C_{k,\ell} - \alpha \sum_{k=1}^{N} C_{k,k}}$$

where

$$\alpha = \begin{cases} 1/2 & \text{if } N = 2 \\ 1 & \text{if } N > 2 \,. \end{cases}$$

And the Confusion Entropy associated to class j (2.3) is redefined as follows:

$$\mathrm{MCEN}_j = - \sum_{k=1, k\neq j}^{N} \left(\widetilde{P}^j_{j,k} \log_{2(N-1)}(\widetilde{P}^j_{j,k}) + \widetilde{P}^j_{k,j} \log_{2(N-1)}(\widetilde{P}^j_{k,j}) \right) ,$$

Finally, a new formula for Confusion Entropy (2.4) is shown:

$$\mathrm{MCEN} = \sum_{j=1}^{N} \widetilde{P}_j \; \mathrm{MCEN}_j \,. \tag{2.6}$$

In both measures *CEN* (2.4) and *MCEN* (2.6) classes can be separated, making it easy to check the effect of modifications in the classifier that affect single classes.

The differences in the values of *CEN* and *MCEN* can be seen in the table bellow. This table shows some simple examples of symmetric and balanced confusion matrices. The problematic characteristic of *CEN* is reflected in the cases where its value surpass *1*.

	$\begin{pmatrix} 6 & 0 \\ 0 & 6 \end{pmatrix}$	$\begin{pmatrix} 5 & 1 \\ 1 & 5 \end{pmatrix}$	$\begin{pmatrix} 4 & 2 \\ 2 & 4 \end{pmatrix}$	$\begin{pmatrix} 3 & 3 \\ 3 & 3 \end{pmatrix}$	$\begin{pmatrix} 2 & 4 \\ 4 & 2 \end{pmatrix}$	$\begin{pmatrix} 1 & 5 \\ 5 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 6 \\ 6 & 0 \end{pmatrix}$
CEN =	0.0000	0.5975	0.8617	1.0000	1.0566	1.0525	1.0000
MCEN =	0.0000	0.5910	0.8000	0.9057	0.9614	0.9891	1.0000

Table 2.3: Examples of CEN and MCEN values of some symmetric and balanced confusion matrices.

Chapter 3

The proposed algorithm: MCENTree

In 2013, Han Jin et al. did not only use *CEN* for measuring the performance of a classifier but also for constructing the classifier itself in *Learning decision trees using Confusion Entropy* [8]. In that work, the authors present the *CENTree*, a decision tree classifier based on the C4.5 algorithm [20]. To construct decision tree algorithms, a splitting criterion must be choosen to determine how the data is divided, from an initial impure node representing the entire data-set to multiple and purer nodes, in a branches and leaves(nodes) tree-like structure.

In the C4.5 algorithm, introduced for the very first time in [21], data is repeatedly divided using *Information Gain* metric as criterion. This metric measures the amount of information an attribute gives about the class, calculating the amount of entropy that is reduced after dividing a node with a certain attribute. The attribute with highest *Information gain* will be used to divide a node.

In the case of the *CENTree*, however, *Confusion entropy* is used to determine the best attribute to split a node. In this work, we propose a modified version of this algorithm that relies in using the *MCEN* as a criterion to iteratively select the most relevant features.

The recursive algorithm, as shown below, starts with the creation of the initial node with the given samples.If all the samples belong to one class the label of the node is set to that class; if, however, there are not more attributes to check, the label is set to the most common class value of the remaining samples.

If the mentioned requirements are not fulfilled, the algorithm does not end. The attribute from the given *Attributes* which best classifies *Samples* is chosen, that is, the one that yields the minimum *MCEN*. For each possible value for that attribute, a branch is added below the root node. Corresponding to each of those values, a new set of *Samples* is formed.

For each of the new sets of *Samples* a sub-tree is added below the new branch. This sub-tree will be formed by the next iteration of the algorithm, with the samples of the corresponding set, and the remaining attributes without the one that has been chosen in the current iteration.

As it is done in [8], a label corresponding to the most common class of a set of samples is set to any sub-set of that set that could be empty due to the lack of instances with a certain attribute value in that set of samples.

Algorithm 1 New Decision Tree with *MCEN* splitting based on [8]

```
 1: MCENTree (Samples, Target attribute, Attributes, Majority Class):
 2: Create a Root node.
 3: if (Samples==empty) then
 4:     Return node with label = Majority Class
 5: else if (all Samples are belonging to one class) then
 6:     return node whose label = classname
 7: else if (Attributes == empty) then
 8:     Return node with label = Majority Class in Samples
 9: else
10:     A = best(Attribute) based on MCEN
11:     for each possible value v(i) of A do
12:         Add a new tree branch below Root where test(A=v(i))
13:         Take subset Samples(v(i)) from Samples with value v(i) for A
14:         Get majority class from Samples.
15:         Add below this new branch the subtree:
16:         MCENTree (Samples(v(i)), Target attribute, Attributes-A,
17:         Majority class)
18:     end for
19: end if
20: return Root
```

Chapter 4

Materials and Methods

In this work the experiments made in *CENTree* have been reproduced to the extent possible, focusing in the binary datasets used in the mentioned article, the ones that are susceptible to have been wrongly classified by the first version of *CEN*. In addition, classification made in four more datasets has been studied. Datasets are available in UCI repository[1]. Selected datasets are the following:

- Tic-tac-toe: Tic-tac-toe Endgame database, created and donated by David W. Aha in 1991. This database contains all the end-game possible configurations of the tic-tac-toe game, assuming that “x” moves first and with “win” for “x” when “x” has achieve a “three-in-a-row”, or “no-win” for “x” as possible classes. Therefore, the data-set contains 958 instances, the number of possible end-game boards, where each entry has nine attributes, corresponding to each board square.
  The class distribution is about 65% for “win” for “x” and 35% for “no-win” for “x”.

- Breast-cancer: Breast cancer data, created by Matjaz Zwitter and Milan Soklic of the Institute of Oncology of Ljubljana and donated by Ming Tan and Jeff Schlimmer in 1988.
  This data-set contains information of 286 patients as different instances, and 9 attributes of both patient and their tumour information plus the class attribute, “no-recurrence-events” with about a 70% of the total number of instances and “recurrence-events” with the remaining 30%.

[1]https://archive.ics.uci.edu/

- Balloons: Balloon databases, collected by Michael Pazzani. In this case a combination of the four different balloons data-set is used, forming a unique data-set of 76 instances with four different attributes of the balloons (color, size, act and age) and the class attribute "inflated", which can be true or false.

- Monks-2: The Monk's Problems: Problem 2. Donated by Sebastian Thrun of the School of Computer Science of Carnegie Mellon University.
  The Monks problems were created to be resolved by any kind of classifier, with the objective of being able to compare the different classifiers to determine which one of them was better. There are three different problems, but in this work only the second problem is used.

- Spect: SPECT heart data, created and donated by Krzysztof J. Cios and Lukasz A. Kurgan of the University of Colorado in 2001.
  This data-set collects heart data from Single Proton Emission Computed Tomography (SPECT) images, and classifies the patients in two different classes, "normal" or abnormal".
  Apart from the class attribute each one of the 267 instances is formed with 22 more binary attributes with features processed from the original SPECT images.

- Congressional voting: 1984 United States Congressional Voting Records Database, data from the Congressional Quarterly Almanac donated by Jeff Schlimmer in 1987.
  This data set includes the votes of all of the 435 United States congressmen (267 democrats, 168 republicans) in 16 different topics (attributes). The class attribute corresponds to the political party to each of the congressmen.

- Chess: Chess End-Game – King+Rook versus King+Pawn on a7. Created by Alen Shapiro and donated by Rob Holte in 1987.
  This data-set include 3196 instances, where each one of them is a board-descriptions for the chess endgame situation. With the 36 attributes the board is described, while with the last attribute, the class attribute, the possibility for the white pieces to win is collected with a "win" or "no-win"" value. The class distribution of the data-set is a 52% of the cases where white can win and 48% where it can not.

- Mushroom: Mushroom Database. Collected from the Mushroom records drawn from The Audubon Society Field Guide to North American Mushrooms (1981) and donated by Jeff Schlimmer in 1987.
  This data-set includes data of 8124 samples of 22 different mushroom species. The data includes 23 attributes for each instance to help

to determine the class attribute, if the mushroom is “edible” or “poisonous”.

In order to reproduce the C4.5 algorithm used as basis of the *CENTree*, a python code created by Barış Can Esmer and found in his GitHub repository [2] is used to create the model. Changing the splitting method of the maximum information gain, used by the C4.5, for a function to calculate and choose the minimum *CEN*, the code for constructing the model of the *CENTree* is created.

With the *CENTree* model constructor codified, a classifier has been programmed to classify test data. For this purpose, the data is splitted into the same percentage as in [8], a 40% of the instances of each data set is used as train data to construct the *CENTree* model, and the remaining 60% of the instances is used as test data to be classified through the model and measure the effectiveness of the classifier.

However, different factors have made the aforementioned experiments not reproducible, and therefore, the results obtained in this work do not match exactly with the exposed in [8]. On the one hand the lack of the original code of a non-fully described modified algorithm, and on the other hand, the lack of information about the seed used in the experiments.

For these reasons the basic classical version of the C4.5 algorithm has been implemented to construct the model, and each data-set has been classified 100 times using 100 different seeds, from seed number 0 to number 99. The results shown in this work are the outcome of calculating the mean of those 100 different results.

Furthermore, more experiments have been made adding noise to the data-sets to study the behaviour of Confusion Entropy in more adverse situations. The analyzed noise levels are of 10%, 20% and 40% and have been applied with the unsupervised attribute filter *”AddNoise”* of Weka. With this filter, a percentage of the given attribute values can be randomly changed. In this case only the class values of the mentioned percentages have been altered, using the seed number 1.

To run all the algorithms required to complete this work a laptop with an Intel Core I7-3517U of 1.90GHz has been used.

[2]https://github.com/barisesmer/C4.5

Chapter 5

Results

5.1 Overall Results

As mentioned in previous sections, the core of this work has been the replication of the *CENTree* algorithm but with *MCEN* instead of *CEN* for the node splitting criterion. Although not being able to faithfully replicate the original experiments due the lack of the originally used algorithm and information about seeds used, in this work four of the datasets used in [8] have been classified though a *CENTree* and a *MCENTree*, in addition to another four different datasets. All the selected datasets are binary, the class of the instances is limited to two different values, where the formula of the Confusion entropy fails.

The results obtained in the first experiments can be seen in the table 5.1, for each dataset, the results obtained with 3 different models are presented, the classic J48, the reproduction of the *CENTree*, and the novel *MCENTree*. For each of these three models, the results obtained with another three different performance measures are shown, *Accuracy*, *CEN* and *MCEN*. Thus, for a good comparison between models and performance measures, the output of the *CENTree* has been measured with both *CEN* and *MCEN*, so as the *MCENTree*. As a reminder, a higher value of accuracy represents a better classification, while in the case of *CEN* and *MCEN* a better classification results in a lower value.

Data-set	Model	Accuracy	CEN	MCEN
Tic-tac-toe	**J48**	0.8120	0.6222	0.6091
	CENTree	0.8098	0.6256	0.6103
	MCENTree	0.8190	0.6072	0.5948
Breast-Cancer	**J48**	0.6454	0.8123	0.7650
	CENTree	0.6247	0.7932	0.7482
	MCENTree	0.6231	0.7877	0.7434
Balloons	**J48**	0.6659	0.8108	0.7456
	CENTree	0.6682	0.7993	0.7339
	MCENTree	0.6666	0.7992	0.7333
Monks-2	**J48**	0.5961	0.8985	0.8311
	CENTree	0.5849	0.8994	0.8337
	MCENTree	0.5815	0.9023	0.8361
Spect	**J48**	0.7626	0.6482	0.6268
	CENTree	0.7695	0.6386	0.6198
	MCENTree	0.7778	0.6279	0.6119
Vote	**J48**	0.9324	0.3182	0.3357
	CENTree	0.9285	0.3307	0.3476
	MCENTree	0.9299	0.3281	0.3458
Chess	**J48**	0.9876	0.0877	0.1004
	CENTree	0.8055	0.6502	0.6335
	MCENTree	0.7897	0.6792	0.6569
Mushroom	**J48**	0.9998	0.0019	0.0022
	CENTree	0.9971	0.0254	0.0301
	MCENTree	0.9977	0.0198	0.0234

Table 5.1: Results for *J48*, *CENTree* and *MCENTree* algorithms.

In this table, slight differences between the three models can be seen. On the one hand J48 achieves better results when being used to classify Monks-2, Vote, Chess and Mushroom in the three analysed performance measures, *Accuracy*, *CEN* and *MCEN*. On the other hand, *MCENTree* yields better results for Tic-tac-toe and Spect for those same three measures, and better results for *CEN* and *MCEN* in Breast-cancer, where J48 still gets the best accuracy, and Balloons, where the best accuracy is given by the *CENTree*.

Comparing *CENTree* with *MCENTree*, the first one gets better results in the three measures in Monks-2 and Chess, while the second one does the same in Tic-tac-toe, Spect, Vote and Mushroom, and obtains better *CEN* and *MCEN* results in Breast-Cancer and Balloons, where *Centree* still manages to get a better accuracy.

Therefore, in the comparison between the *CENTree* and *MCENTree*, the second one yields better results in most of the studied cases. If the three models are compared, J48 and *MCENTree* get the best results in a similar number of cases, J48 for Monks-2, Vote, Chess and Mushroom, and *MCETree* for Tic-tac-toe, Breast-cancer, Balloons and Spect.

A closer look to the results can be taken with the table 5.2, which shows the worst results obtained among the 100 times the classifier has been run.

In this table of worst results, and the ones in the same line for each different noise levels, the five worst results of the *CENTree* have been collected. For each of these values, corresponding to a classification made with a specific seed, the values for the classification made with the same seed but with the *MCENTree* are shown, plus the results of measuring the confusion matrix given by the *CENTree* with *MCEN*.

5.2 Critical Cases

We have seen how *MCENTree* yields better results than *CENTree*, but still have not seen how the formula of Confusion entropy fails and presents values over the [0,1] range. To be able to see that undesired behaviour, additional experiments have been done in this work. In these experiments, with the addition of noise, the performance of the studied models is tested in exceptional adverse situations. These experiments have been done with noise levels of 10%, 20% and 40% of the total number of entries of the datasets.

In table 5.3 the results of the classifications made with a level of 10% of noise can be seen. As expected, with the addition of noise, obtained accuracy, *CEN* and *MCEN* values are worse than those obtained with the clean data. Also, it can be appreciated how the values of *CEN* tend to increase more than the ones of the *MCEN* do, but still don't surpass the theoretical limit of 1.

Dataset	CENTree	CENTree-MCEN	MCENTree
Tic-tac-toe	0.7286	0.6972	0.6686
	0.7142	0.6852	0.6298
	0.7090	0.6783	0.6015
	0.7041	0.6779	0.6317
	0.7034	0.6745	0.6609
Breast-cancer	0.9412	0.8702	0.8728
	0.9159	0.8531	0.8620
	0.9055	0.8450	0.8365
	0.8944	0.8319	0.8090
	0.8894	0.8289	0.8213
Balloons	0.9513	0.8625	0.8431
	0.9441	0.8622	0.8622
	0.9411	0.8605	0.8605
	0.9260	0.8397	0.8397
	0.9260	0.8397	0.8397
Monks-2	0.9438	0.8584	0.8650
	0.9425	0.8667	0.8681
	0.9385	0.8643	0.8653
	0.9363	0.8640	0.8726
	0.9363	0.8663	0.8745
Spect	0.7281	0.6935	0.6215
	0.7195	0.6844	0.6885
	0.7107	0.6832	0.6566
	0.7106	0.6848	0.6562
	0.7087	0.6843	0.6421
Vote	0.4449	0.4441	0.3749
	0.4404	0.4506	0.4257
	0.4256	0.4391	0.3931
	0.4253	0.4388	0.4503
	0.4226	0.4319	0.3854
Chess	0.6923	0.6678	0.6765
	0.6865	0.6640	0.6696
	0.6854	0.6630	0.6894
	0.6844	0.6616	0.6779
	0.6842	0.6617	0.6664
Mushroom	0.0511	0.0599	0.0528
	0.0498	0.0585	0.0142
	0.0490	0.0574	0.0494
	0.0467	0.0549	0.0505
	0.0467	0.0549	0.0549

Table 5.2: 5 worst Confusion Entropy results out of 100 executions. Data without noise.

Data-set	Model	Accuracy	CEN	MCEN
Tic-tac-toe	**J48**	0.6808	0.8276	0.7732
	CENTree	0.6840	0.8216	0.7655
	MCENTree	0.6830	0.8214	0.7647
Breast-Cancer	**J48**	0.5711	0.8974	0.8322
	CENTree	0.5597	0.8617	0.8025
	MCENTree	0.5596	0.8567	0.7985
Balloons	**J48**	0.5653	0.8896	0.8049
	CENTree	0.5666	0.8911	0.8070
	MCENTree	0.5673	0.8917	0.8079
Monks-2	**J48**	0.5643	0.9312	0.8553
	CENTree	0.5631	0.9231	0.8497
	MCENTree	0.5591	0.9265	0.8525
Spect	**J48**	0.5956	0.9105	0.8335
	CENTree	0.6007	0.8947	0.8190
	MCENTree	0.6013	0.8960	0.8203
Vote	**J48**	0.7918	0.6643	0.6427
	CENTree	0.8037	0.6366	0.6179
	MCENTree	0.8039	0.6379	0.6196
Chess	**J48**	0.7975	0.6674	0.6482
	CENTree	0.6990	0.8185	0.7661
	MCENTree	0.6919	0.8276	0.7731
Mushroom	**J48**	0.7984	0.6665	0.6477
	CENTree	0.8076	0.6471	0.6311
	MCENTree	0.8073	0.6485	0.6326

Table 5.3: Results with a level of 10% of noise.

But as it has been said before, the results shown in these tables are the mean of the results of the 100 executions of the algorithm. If we take a closer look to the results, looking for the worst ones among all of them (table 5.4), the anomalous behaviour of the *CEN* is reflected in the results obtained for Balloons dataset.

Dataset	CENTree	CENTree-MCEN	MCENTree
Tic-tac-toe	0.8835	0.8147	0.8076
	0.8739	0.8097	0.8097
	0.8717	0.7989	0.7908
	0.8694	0.8049	0.8101
	0.8649	0.8011	0.7900
Breast-cancer	0.9578	0.8895	0.8651
	0.9573	0.8780	0.8799
	0.9555	0.8813	0.8758
	0.9505	0.8773	0.8802
	0.9498	0.8805	0.8536
Balloons	1.0229	0.9314	0.9314
	1.0199	0.9228	0.9228
	1.0075	0.9153	0.9153
	0.9988	0.9046	0.9046
	0.9976	0.9043	0.9043
Monks-2	0.9868	0.9009	0.9009
	0.9731	0.8896	0.8905
	0.9624	0.8819	0.8784
	0.9613	0.8808	0.8708
	0.9558	0.8764	0.8780
Spect	0.9903	0.8976	0.8019
	0.9698	0.8828	0.8604
	0.9671	0.8815	0.8815
	0.9668	0.8797	0.8828
	0.9627	0.8747	0.8564
Vote	0.7632	0.7239	0.7184
	0.7474	0.6960	0.6906
	0.7373	0.7041	0.6696
	0.7281	0.6971	0.6727
	0.7259	0.6916	0.6628
Chess	0.8525	0.7907	0.7852
	0.8500	0.7911	0.7933
	0.8480	0.7897	0.7811
	0.8463	0.7882	0.7788
	0.8456	0.7868	0.7908
Mushroom	0.6739	0.6538	0.6398
	0.6714	0.6505	0.6459
	0.6696	0.6494	0.6369
	0.6693	0.6496	0.6457
	0.6686	0.6496	0.6400

Table 5.4: 5 worst Confusion Entropy results out of 100 executions. Data with level of 10% of noise.

Going further with the experiment, the results obtained with a level of 20% of noise are shown in table 5.5. In the same way as before, in this case the results of accuracy, *CEN* and *MCEN* are worse than the ones obtained with a level of noise of a 10%. We can see how the gap between the values of *CEN* and *MCEN* increases as the noise level increases, but in these overall results *CEN* still manages to cover its wrong behaviour.

Data-set	Model	Accuracy	CEN	MCEN
Tic-tac-toe	**J48**	0.5926	0.9227	0.8458
	CENTree	0.5918	0.9185	0.8382
	MCENTree	0.5936	0.9159	0.8356
Breast-Cancer	**J48**	0.5338	0.9380	0.8600
	CENTree	0.5169	0.9193	0.8444
	MCENTree	0.5159	0.9152	0.8414
Balloons	**J48**	0.5228	0.9171	0.8260
	CENTree	0.5219	0.9182	0.8274
	MCENTree	0.5242	0.9181	0.8276
Monks-2	**J48**	0.5358	0.9595	0.8762
	CENTree	0.5467	0.9457	0.8652
	MCENTree	0.5445	0.9468	0.5467
Spect	**J48**	0.5581	0.9400	0.8551
	CENTree	0.5501	0.9387	0.8537
	MCENTree	0.5500	0.9387	0.8533
Vote	**J48**	0.6656	0.8489	0.7880
	CENTree	0.6732	0.8347	0.7758
	MCENTree	0.6708	0.8374	0.7777
Chess	**J48**	0.6665	0.8605	0.7988
	CENTree	0.6117	0.9148	0.8392
	MCENTree	0.6095	0.9168	0.8406
Mushroom	**J48**	0.6665	0.8610	0.7993
	CENTree	0.6661	0.8581	0.7959
	MCENTree	0.6685	0.8565	0.7951

Table 5.5: Results with a level of 20% of noise.

In this case, however, more results over the [0,1] range can be seen if we analyze the worst results (table 5.6) obtained. Four of the results of the classification made with the Spect dataset surpass the upper value of the theoretical range, as well as one for Monks-2 dataset and, at least, the five worst results of the classification done to the Balloons dataset surpass the maximum value of the range.

Dataset	CENTree	CENTree-MCEN	MCENTree
Tic-tac-toe	0.9573	0.8736	0.8677
	0.9548	0.8686	0.8609
	0.9541	0.8702	0.8606
	0.9538	0.8680	0.8528
	0.9531	0.8684	0.8366
Breast-cancer	0.9958	0.9088	0.9129
	0.9941	0.9147	0.8863
	0.9874	0.9074	0.9066
	0.9867	0.9035	0.9043
	0.9862	0.8978	0.9017
Balloons	1.0577	0.9654	0.9654
	1.0456	0.9472	0.9020
	1.0402	0.9450	0.9450
	1.0402	0.9450	0.9450
	1.0369	0.9389	0.9389
Monks-2	1.0006	0.9076	0.9118
	0.9869	0.8974	0.9083
	0.9853	0.8972	0.8900
	0.9850	0.8980	0.8991
	0.9836	0.8966	0.8984
Spect	1.0172	0.9200	0.9164
	1.0092	0.9149	0.9097
	1.0020	0.9091	0.9139
	1.0005	0.9067	0.8967
	0.9993	0.9010	0.8856
Vote	0.9147	0.8407	0.8398
	0.9123	0.8390	0.8239
	0.8976	0.8242	0.8488
	0.8966	0.8236	0.8107
	0.8904	0.8194	0.8272
Chess	0.9411	0.8598	0.8598
	0.9394	0.8585	0.8557
	0.9338	0.8540	0.8538
	0.9320	0.8530	0.8489
	0.9317	0.8522	0.8576
Mushroom	0.8756	0.8097	0.8060
	0.8715	0.8070	0.8037
	0.8706	0.8055	0.8036
	0.8701	0.8059	0.8045
	0.8699	0.8046	0.8093

Table 5.6: 5 worst Confusion Entropy results out of 100 executions. Data with a level of 20% of noise.

With a level of 40% of noise, all the cases measured with *CEN* are close to a value of 1 (table 5.7), what would indicate a total misclassification, while the values for *MCEN* are about a 0.1 lower than the ones of the *CEN*. The values for accuracy are around 0.5, showing that only around a 50% of the elements have been correctly classified.

Data-set	Model	Accuracy	CEN	MCEN
Tic-tac-toe	**J48**	0.5072	0.9905	0.8972
	CENTree	0.5033	0.9824	0.8869
	MCENTree	0.5023	0.9802	0.8843
Breast-Cancer	**J48**	0.4824	0.9901	0.8958
	CENTree	0.4776	0.9818	0.8884
	MCENTree	0.4801	0.9748	0.8821
Balloons	**J48**	0.4555	0.9632	0.8672
	CENTree	0.4579	0.9636	0.8679
	MCENTree	0.4573	0.9659	0.8703
Monks-2	**J48**	0.4924	0.9934	0.8997
	CENTree	0.5038	0.9834	0.8912
	MCENTree	0.5050	0.9834	0.8913
Spect	**J48**	0.5014	0.9821	0.8885
	CENTree	0.5036	0.9758	0.8825
	MCENTree	0.5001	0.9785	0.8847
Vote	**J48**	0.5150	0.9780	0.8860
	CENTree	0.5205	0.9698	0.8782
	MCENTree	0.5168	0.9733	0.8813
Chess	**J48**	0.5218	0.9859	0.8942
	CENTree	0.5116	0.9898	0.8968
	MCENTree	0.5125	0.9894	0.8965
Mushroom	**J48**	0.5194	0.9880	0.8960
	CENTree	0.5125	0.9866	0.8936
	MCENTree	0.5123	0.9889	0.8960

Table 5.7: Results with a level of 40% of noise.

Under this extreme conditions, however, the atypical performance of CEN is noticeable in all the studied datasets but one when looking the individual results, what can be seen in table 5.8. For six out of the eight datasets, at least the five worst results surpass the upper limit of the [0,1] range, Tic-tac-toe, Breast-Cancer, Ballons, Monks-2, Spect and Vote. For Chess dataset, only the worst four results surpass the limit. And finally, Mushrooms dataset still manages to obtain results under the upper limit of the range.

Dataset	CENTree	CENTree-MCEN	MCENTree
Tic-tac-toe	1.0123	0.9158	0.9231
	1.0120	0.9145	0.9015
	1.0096	0.9131	0.9160
	1.0087	0.9122	0.9129
	1.0080	0.9117	0.9211
Breast-cancer	1.0331	0.9341	0.9231
	1.0297	0.9333	0.9285
	1.0290	0.9319	0.9357
	1.0273	0.9337	0.9138
	1.0264	0.9310	0.9360
Balloons	1.0606	0.9708	0.9708
	1.0486	0.9538	0.9193
	1.0479	0.9498	0.9498
	1.0402	0.9450	0.9450
	1.0396	0.9470	0.9470
Monks-2	1.0179	0.9205	0.9206
	1.0155	0.9191	0.9201
	1.0136	0.9178	0.9108
	1.0133	0.9188	0.9082
	1.0129	0.9177	0.9161
Spect	1.0389	0.9414	0.9446
	1.0308	0.9317	0.9192
	1.0275	0.9288	0.9316
	1.0243	0.9263	0.9215
	1.0243	0.9263	0.9189
Vote	1.0158	0.9191	0.9207
	1.0069	0.9104	0.9155
	1.0024	0.9084	0.9084
	1.0017	0.9067	0.9067
	1.0001	0.9036	0.9074
Chess	1.0084	0.9124	0.9086
	1.0075	0.9116	0.9084
	1.0038	0.9085	0.9044
	1.0018	0.9070	0.9046
	0.9991	0.9042	0.9031
Mushroom	0.9988	0.9047	0.9084
	0.9985	0.9042	0.9038
	0.9968	0.9021	0.8996
	0.9961	0.9020	0.9013
	0.9948	0.9011	0.9006

Table 5.8: 5 worst Confusion Entropy results out of 100 executions. Data with a level of 40% of noise.

In all tables of worst results that have been shown in this work, it can be observed how all the individual cases where a classification made with the *CENTree* algorithm that surpassed the theoretical range of values show their values corrected when being classified using the same seed with the *MCENTree* algorithm.

This can be easily seen in the following graphs, formed with the mean vales of the worst results for each of the studied noise levels.

Next figures show results for both *CENTree* and *MCENTree* for each database, with no noise and with the analyzed levels of 10%, 20% and 40% of noise. It can be appreciated how *CEN* exceeds the value of "1" when noise is increased. For Tic-tac-toe, Breast-cancer, Monks-2, Vote and Chess the limit is only surpassed when reaching a level of noise of 40%. Spect dataset surpass the limit with a level of noise of 20%, and Balloons dataset only needs a level of noise of 10% to show this behaviour. Lastly, studied noise levels are not high enough to make the results of Mushrooms dataset surpass the limit of "1".

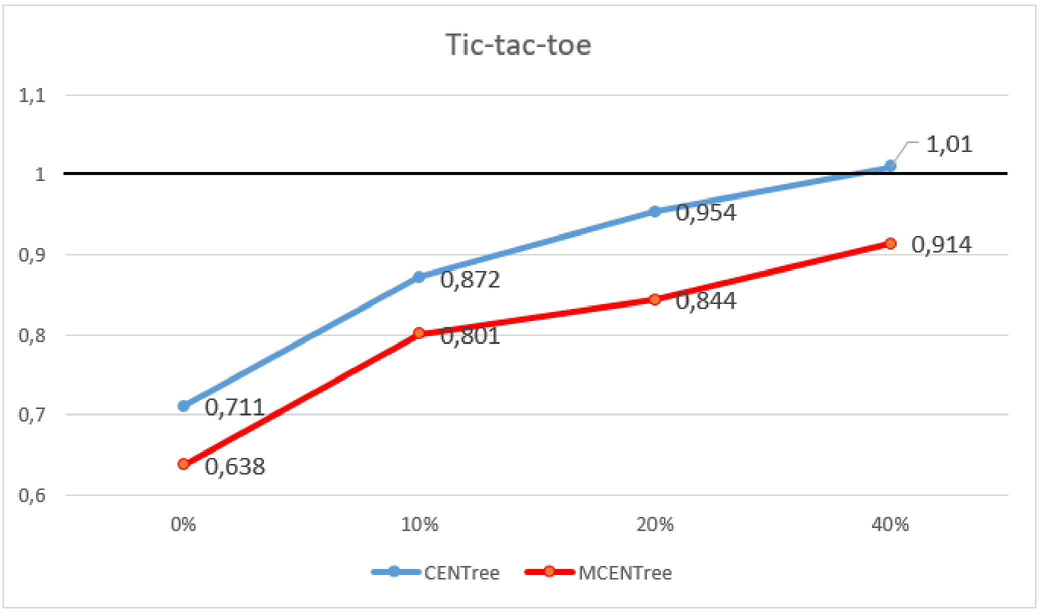

Figure 5.1: Graph with mean values of the worst results obtained for Tic-tac-toe dataset.

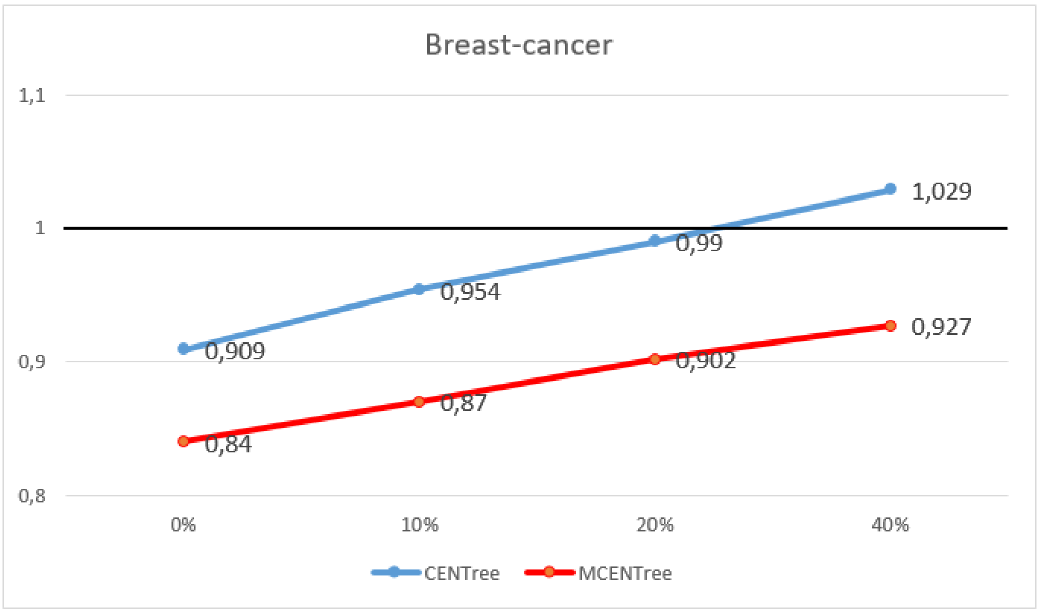

Figure 5.2: Graph with mean values of the worst results obtained for Breast-cancer dataset.

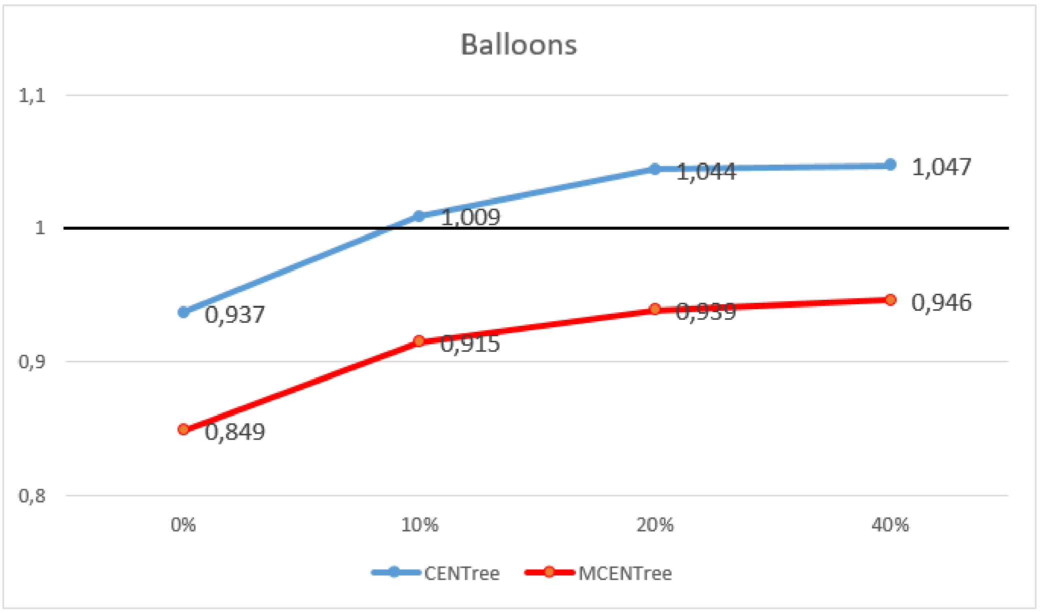

Figure 5.3: Graph with mean values of the worst results obtained for Balloons dataset.

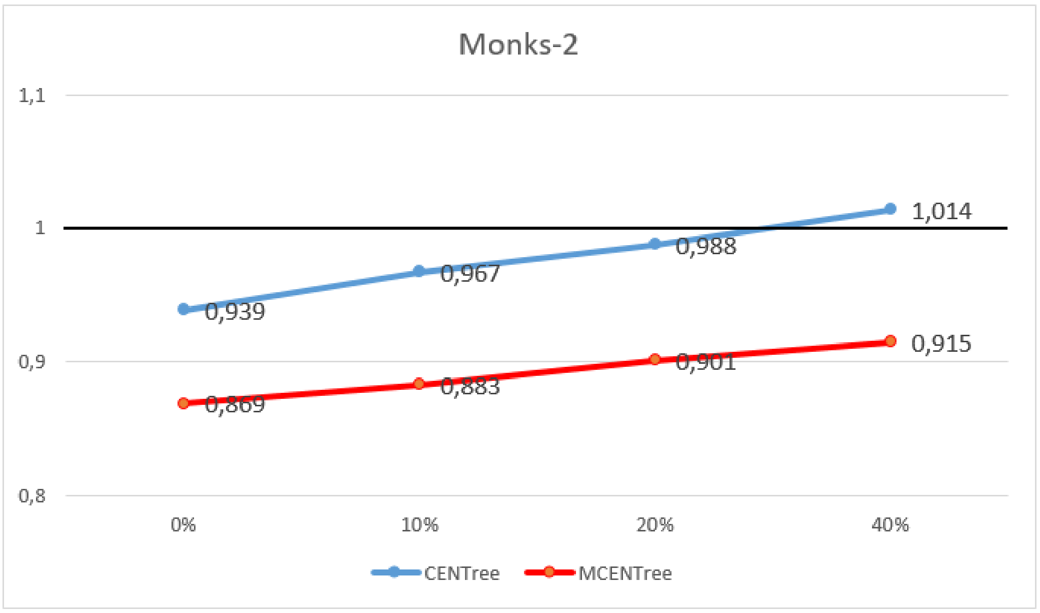

Figure 5.4: Graph with mean values of the worst results obtained for Monks-2 dataset.

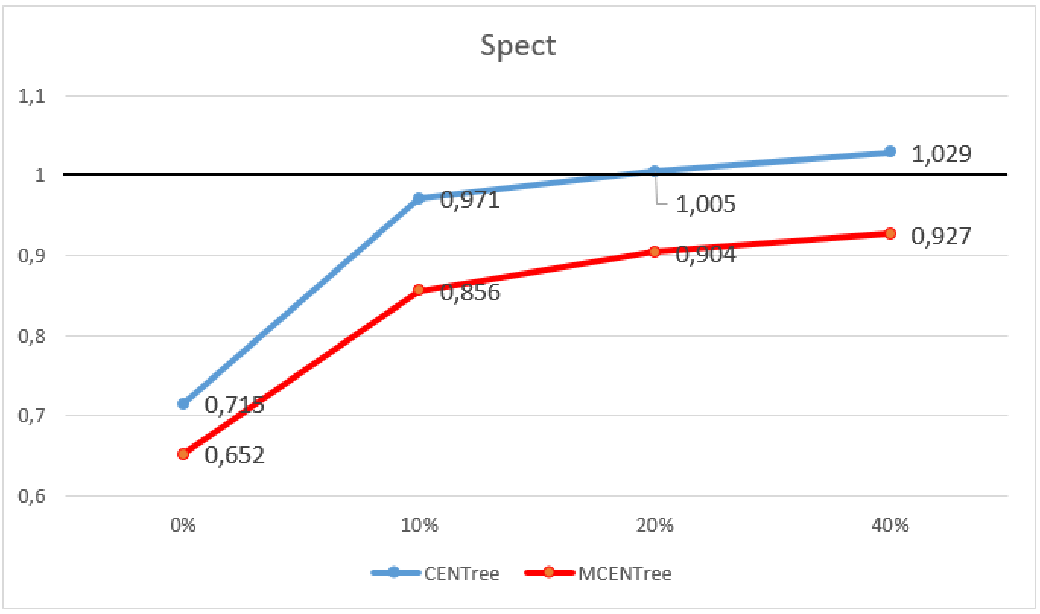

Figure 5.5: Graph with mean values of the worst results obtained for Spect dataset.

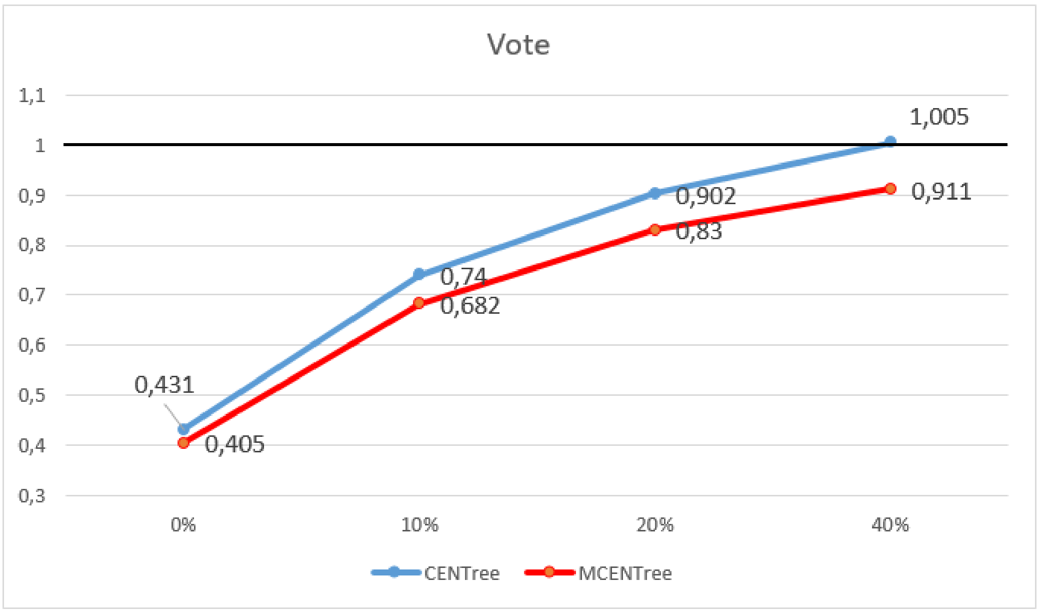

Figure 5.6: Graph with mean values of the worst results obtained for Vote dataset.

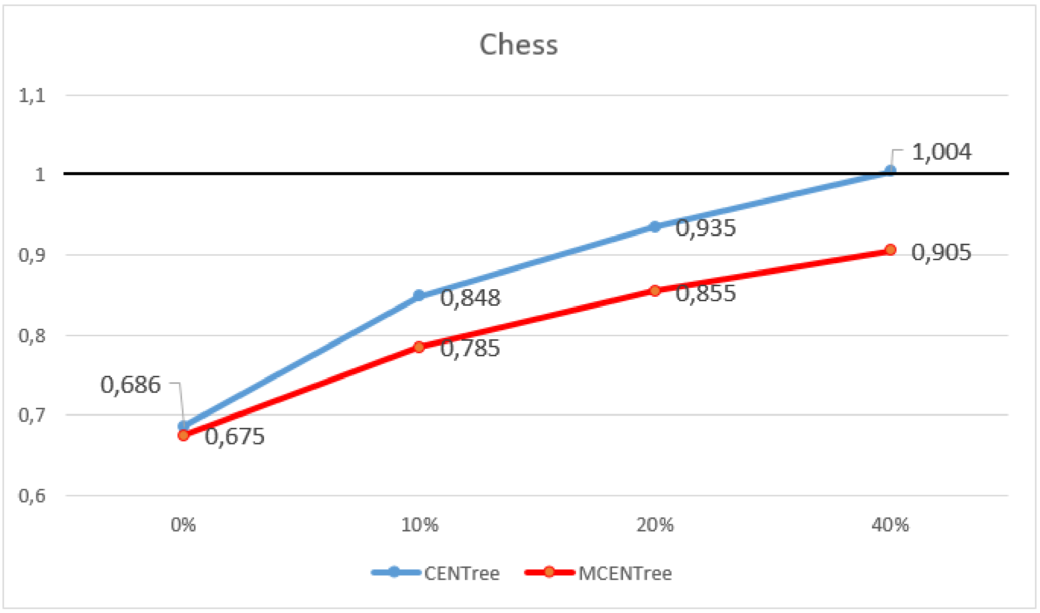

Figure 5.7: Graph with mean values of the worst results obtained for Chess dataset.

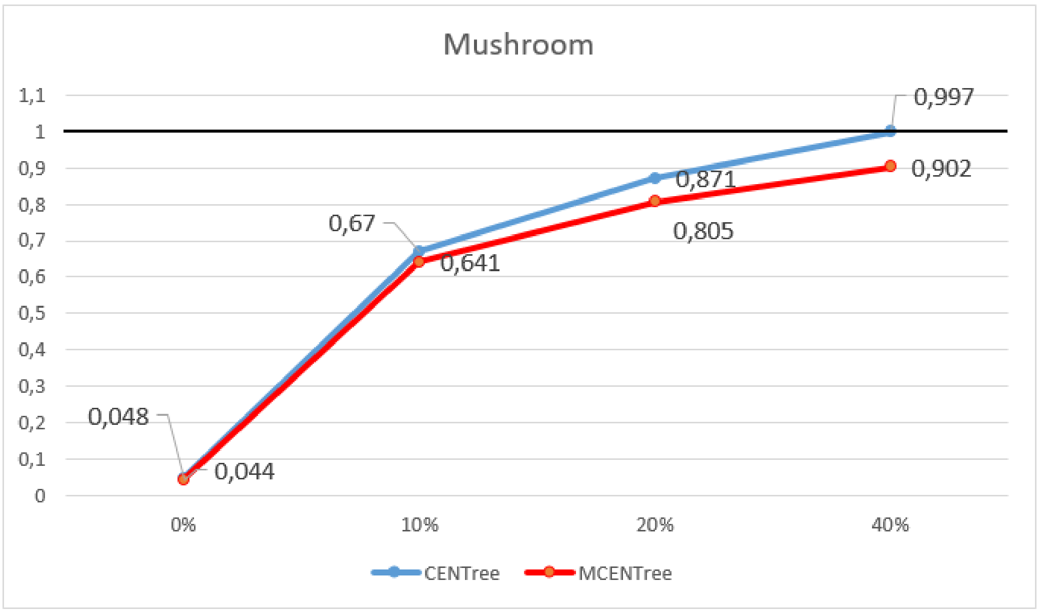

Figure 5.8: Graph with mean values of the worst results obtained for Mushrooms dataset.

Chapter 6

Conclusions and future work

6.1 Conclusions

In this work experiments regarding the *CEN* based decision tree algorithm, *CENTree*, have been conducted. In order to do this, first, the algorithm has been replicated to the extent possible, and a classifier has been constructed. The data-sets selected to be classified have been four different binary data-sets used in [8] plus another four binary data-sets, all of them from the same UCI repository mentioned before. To test the effectiveness of the classification algorithm the datasets have been modified with different levels of noise to reproduce adverse situations.

For the reproduction of the algorithm, the C4.5 algorithm has been used as base, while the node splitting method has been changed from the classical information gain based criterion to a *CEN* based pruning criterion (the *CENsplit* method). The lack of the original code used in [8] has been the first issue in this work, as the configuration of the algorithm used is unknown and techniques like the so called Minimun-CEN-Pruning, a method to prevent overfitting, are not explained in detail.

On the other hand, the lack of information about the seeds used in the original work has represented the second problem to replicate the results of the experiments. To handle this issue trying to avoid biased results, 100 seeds, from number 0 to 99, have been used to obtain 100 different outcomes from the data classification. The results presented in this work are the mean of the different results.

Despite the mentioned difficulties a reliable *CENTree* has been constructed, and the *MCENTree*, replacing the old definition of *CEN* with the new one, *MCEN*. While in the beginning the results of the classifications made through both *CENTree* and *MCENTree* may seem similar, the differences become more and more noticeable as higher levels of noise are introduced in the datasets, overloading the resulting confusion matrices outside the main diagonal, to the point where the *CENTree* exposes its malfunctioning with results over the [0,1] range.

Therefore, with the obtained results, it can be concluded that the substitution of *CEN* by *MCEN* yields better results in this particular case of classification made with a decision tree based on Confusion Entropy.

6.2 Future work

In this work another research has been replicated to make an enhancement concerning to the Confusion Entropy definition and compare the obtained results with the originals. However, the difficulties to replicate the original experiments have made it impossible to get the same results as the ones obtained in the original work. A possible future goal would be to recreate the experiments under the original circumstances, in case of being able to obtain the original algorithm and know the seeds used. In spite of this, the conclusions of the comparison would not presumably be different.

In the same line, as future research goals, more works from the literature in which the concept of Confusion Entropy has been used can be replicated. Thus, *CEN* will be replaced with *MCEN* to make comparisons in the results between these and other performance measures like *Accuracy* or *Matthews Correlation Coefficient (MCC).*

In this way, the improvement presented by *MCEN* with respect to *CEN* could be concluded by empirical experimentation.

Taking a step further, looking for new original applications where *MCEN* could be useful can always be considered.

Bibliography

[1] Daniel T Larose and Chantal D Larose. *Discovering knowledge in data: an introduction to data mining*. John Wiley & Sons, 2014.

[2] B.W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442 – 451, 1975.

[3] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 7 1948.

[4] Jin-Mao Wei, Xiao-Jie Yuan, Qing-Hua Hu, and Shu-Qin Wang. A novel measure for evaluating classifiers. *Expert Systems with Applications*, 37(5):3799–3809, 2010.

[5] Giuseppe Jurman and Cesare Furlanello. A unifying view for performance measures in multi-class prediction. *arXiv preprint arXiv:1008.2908*, 2010.

[6] Rosario Delgado and J David Núñez-González. Enhancing confusion entropy as measure for evaluating classifiers. In *The 13th International Conference on Soft Computing Models in Industrial and Environmental Applications*, pages 79–89. Springer, 2018.

[7] Rosario Delgado and J David Núñez-González. Enhancing confusion entropy (cen) for binary and multiclass classification. *PloS one*, 14(1):e0210264, 2019.

[8] Han Jin, Xiao-Ning Wang, Fei Gao, Jian Li, and Jin-Mao Wei. Learning decision trees using confusion entropy. In *Machine Learning and Cybernetics (ICMLC), 2013 International Conference on*, volume 2, pages 560–564. IEEE, 2013.

[9] Foster J Provost, Tom Fawcett, Ron Kohavi, et al. The case against accuracy estimation for comparing induction algorithms. In *ICML*, volume 98, pages 445–453, 1998.

[10] Giuseppe Jurman, Samantha Riccadonna, and Cesare Furlanello. A comparison of mcc and cen error measures in multi-class prediction. *PloS one*, 7(8):e41882, 2012.

[11] Yazan F Roumani, Jerrold H May, David P Strum, and Luis G Vargas. Classifying highly imbalanced icu data. *Health care management science*, 16(2):119–128, 2013.

[12] Madhav Sigdel and Ramazan Savas Ayg¨un. Pacc-a discriminative and accuracy correlated measure for assessment of classification results. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 281–295. Springer, 2013.

[13] Nader Salari, Shamarina Shohaimi, Farid Najafi, Meenakshii Nallappan, and Isthrinayagy Karishnarajah. A novel hybrid classification model of genetic algorithms, modified k-nearest neighbor and developed backpropagation neural network. *PLOS one*, 9(11):e112987, 2014.

[14] Maria Martins, Cristina Santos, Lino Costa, and Anselmo Frizera. Feature reduction with pca/kpca for gait classification with different assistive devices. *International Journal of Intelligent Computing and Cybernetics*, 8(4):363–382, 2015.

[15] Jr´´mie Sublime, Basarab Matei, Gun´a¨el Cabanes, Nistor Grozavu, Youn`es Bennani, and Antoine Cornu´ejols. Entropy based probabilistic collaborative clustering. *Pattern Recognition*, 72:144–157, 2017.

[16] Francisco Antunes, Bernardete Ribeiro, and Francisco Pereira. Probabilistic modeling and visualization for bankruptcy prediction. *Applied Soft Computing*, 60:831–843, 2017.

[17] Rub´en Jerves-Cobo, Gonzalo C´rdova-Vela, Xavier I˜iguez-Vela, Catalina Dáz-Granda, Wout Van Echelpoel, Felipe Cisneros, Ingmar Nopens, and Peter LM Goethals. Model-based analysis of the potential of macroinvertebrates as indicators for microbial pathogens in rivers. *Water*, 10(4):375, 2018.

[18] Yazan F Roumani, Yaman Roumani, Joseph K Nwankpa, and Mohan Tanniru. Classifying readmissions to a cardiac intensive care unit. *Annals of Operations Research*, 263(1-2):429–451, 2018.

[19] Sepand Haghighi, Masoomeh Jasemi, Shaahin Hessabi, and Alireza Zolanvari. PyCM: Multiclass confusion matrix library in Python. *Journal of Open Source Software*, 3(25):729, 2018.

[20] J Ross Quinlan. *C4. 5: programs for machine learning.* Elsevier, 2014.

[21] Steven L Salzberg. C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16(3):235–240, 1994.