HyCASTLE: a Hybrid ClAssification System based on Typicality, Labels and Entropy.

Michele Delli Veneri[a], Antonio Picariello[a,e], Stefano Cavuoti[b,c], Massimo Brescia[b], Giancarlo Sperlì[a], Vincenzo Moscato[a], Roberto Abbruzzese[f,g], Giuseppe Longo[c,d]

[a]*Department of Information Technology and Electrical Engineering (DIETI), University of Naples Federico II, Via Claudio 21, 80125, Italy.*
[b]*INAF - Astronomical Observatory of Capodimonte, Salita Moiariello 16, I-80131 Napoli, Italy*
[c]*Department of Physics "E. Pancini", University of Naples Federico II, via Cintia, 21, I-80126 Napoli, Italy*
[d]*Napoli Unit of the INFN, via Cinthia, 21, I-80126 Napoli, Italy*
[e]*CINI - ITEM National Lab, Complesso Universitario Monte S.Angelo, Naples, Italy*
[f]*Eustema S.p.A., Via G. Porzio,4, 80143 Napoli, Italy*
[g]*Dipartimento Scienze Aziendali - Management & Innovation Systems, University of Salerno*

Abstract

Traditional supervised classification models aim to approximate the functional mapping between instance attributes and their class labels. These models, however, do not consider the interdependence between instances and global characteristics of data and thus often they lead to poor classification results. In this work, we present a novel hybrid classification model – named HyCASTLE – designed to solve the main shortcomings of hybrid models: they make hypotheses on the underlying data distribution and do not consider the effect of noise. HyCASTLE utilizes a non-parametric estimator to capture the underlying data distribution and creates entirely data-driven shape-free clusters. HyCASTLE then refines this cluster configuration using both data topology and available labels through an iterative cluster aggregation and separation process. We evaluated HyCASTLE performance on 35 datasets and compare it with both traditional and hybrid classification models. Our results show that HyCASTLE has comparable or better performance than the other models and results to be more resilient to class noise.

Keywords: machine learning, hybrid models, multi-class classification, supervised classification, clustering

1. Introduction

In the last few decades, several algorithmic solutions have been proposed to tackle the problem of classification and, to date, most of them are supervised learning approaches. In such a context, class labels for the training set are known, and the goal of the algorithms is to approximate the functional mapping between the sample attributes and the class labels. When the mapping is known, it can be used to predict the class labels for unseen samples. Although these algorithms have often shown good performances on a plethora of applications, they do not consider the inherent structure of the data when constructing the modelling and assume that the instances are drawn from independent distributions, thus ignoring all possible inter-dependencies between instances themselves[1]. Unsupervised learning, on the contrary, refers to the problem of identifying groups of instances that share common properties and can be used to map inter-dependencies within the data. In recent years, there have been several attempts to build hybrid approaches combining unsupervised clustering and classification through the intuition that unsupervised clustering methods group instances on the basis of their mutual relationships and thus can provide constraints suitable for a classification algorithms [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12].

1.1. Literature Review

The usual way of constructing such hybrid models is to combine a clustering algorithm, most of the time density or graph-based, with a common supervised classification algorithm such as Iterative Dichotomiser 3 (ID3 Breiman et al. 13) , C4.5 [14], multi-interval ID3 (M-I ID3) [15], k-nearest neighbours (KNN) [16], Support Vector Machines (SVM) [17], naive Bayes [18], logistic regression (LR) [19] and neural networks [20]. More recently, Deep Learning has also been applied with some moderate success on structured data but, usually, good performance with this type of models is achieved through extensive training in data-rich environments. Hence, they suffer from a limited range of applications and a high computational cost.

Archaya et al. [21] proposed to use clustering as a post-processing technique, in order to refine the output obtained by a classifier, while Chakraborty et al. [10] combined multiple clustering and classification methods using a convex optimization function. Rajamohamed et al. [8] merged the clustering obtained through a K-mean, with five supervised classification models. The best results were achieved with the use of a Support Vector Machine as classifier (RK+SVM). Gaddam et al. proposed a hybrid model called K-means+ID3 [9], which first separates the data into K disjoint clusters and then trains an ID3 decision tree on each cluster. Bertini et al. proposed a method called Attribute-based Decision Graph (AbDG, Bertini et al. 12) for constructing a new type of graph, called Attribute-based Decision Graph. In particular, given a vector-based data set, an AbDG is built by partitioning each data attribute range into disjoint intervals and representing each interval as a vertex. The edges are then established between vertices from different attributes according to a pre-defined pattern. Classification is performed through a matching process among the attribute values of the new instance and AbDG. Bose et al. proposed a two-stage hybrid model called KM-Boosted C5.0 [5], composed by an unsupervised clustering algorithm and a boosted C5.0 algorithm. Kaechinport et al. proposed a novel hybrid model called tree bagging and weighted clustering (TBWC) [6] in which important attributes and their weights are found by applying a decision tree bagging and then this weighted attributed are used to generate clusters with which new instances are classified. Ma et al. proposed a novel hybrid algorithm called spectral clustering and deep neural network (SCDNN) [11] in which spectral clustering is combined with Deep Neural Networks (DNN). Their algorithm first proceeds by dividing the training set into K clusters, for which they compute the centroids and then by training a DNN on each of these clusters. Afterwards, by using a similarity criterion, the test set is divided into K clusters, and each subset is processed by the most appropriate DNN. Finally, Xiao et al. proposed a hybrid classification framework based on clustering (HCFC) [7], which firstly applies a clustering algorithm to partition the training samples in K clusters and then constructs a clustering-based attribute selection measure (hybrid information gain ratio), based upon a C4.5 [14] decision tree training. As a clustering algorithm, they chose to test both an improved version of the K-means algorithm [22] and DBSCAN algorithm [23].

1.2. Our Motivations

All cited authors in Sec. 1.1 provided useful insights in the development of hybrid models scenario, however, we can trace some common shortcomings for such models:

- they are mostly based on clustering algorithms that make assumptions on the underlying data distribution, thus resulting in predefined numbers of clusters (in the case of K-means-like approaches) or in clusters with an imposed shape;
- they all have well separated clustering and classification phases. They start with clustering and then, on the basis of the extracted information, perform the classification. However, they do not use information acquired during classification to update their clustering results;
- with the exception of [7], they do not consider the effect of noise on the performances of the implemented methods. A limitation which prevents their usage in many fields.

1.3. Our Contribution

We propose a novel method named Hybrid ClAssification System based on Typicality, Labels and Entropy (HyCASTLE), which makes use of the *typicality*, firstly introduced by Angelov et al. [24], to find the initial clustering and maps the training set onto a cluster configuration, in which the number,size and shape of the clusters are dictated by the data topology in an unsupervised approach.

The algorithm then proceeds by applying a cluster aggregation/separation strategy recursively, until it reaches a minimum in the cluster's configuration entropy. During aggregation, the algorithm computes the cluster's centroids distance distribution (D_C), and separates the clusters, based on their member's labels, in a number of groups equals to the number of classes observed in the data plus one containing multi-class clusters. A graph is constructed on each group by selecting clusters centroids as nodes, and their cluster Gini index as an attribute. In groups containing pure clusters, an edge is created between two given nodes if their mutual distance is shorter than the mean of D_C multiplied by a user-defined coefficient α (*distance based threshold*). At the same time, in the group containing spurious clusters an edge is created if the two clusters meet the distance-based threshold and if their aggregation minimizes the Gini index of the cluster configuration. Then the algorithm finds all sets of connected components in each group and aggregates them. The cluster's centroids distance distribution is then re-computed, and the aggregation process goes on until no edges can be created. During separation, a Classification and Regression Tree (CART) model [13] is trained on each cluster and divides it based on its information content. Each cluster is then replaced by a number of sub-clusters equaling the number of leaf nodes produced by the tree. In the prediction phase, the algorithm computes the probability that a given instance may belong to all cluster present in the cluster configuration found after the training stage. Then, in order to compute class probabilities, all probabilities of single-class clusters, i.e. pure clusters, are added together to form the respective class probability. Instead, the probabilities relative to multi-class clusters, i.e. spurious clusters, are divided among class probabilities, proportionally to each class frequency found in the clusters. The experiments performed on 35 UCI (University of California Irvine) [25] datasets show that our model achieves better or comparable results than those obtained with the models proposed by Bertini et al. [12] and Xiao et al. [7] and several others hybrid and pure classification models. The main novelties of HyCASTLE can be summarized as follows:

1. the algorithm does not make any prior assumption on the underlying probability distribution of the data and therefore produces a cluster configuration whose members are free to take any given shape. Moreover, it can identify outliers in the training set and eliminate them in order to avoid their effect on the measure of the topological proprieties of the data;
2. we propose a hybrid instance assignment method that takes into account both the topological properties of the data and its labels to refine clusters. This is done in order to create clusters through a compromise between the unknown truth (the topological structure of the data) and the known (labels);
3. an optimization loop updates the clustering on the basis of the classification results, in order to minimize the cluster configuration entropy. This search is performed with a dynamical topological constrain;
4. the method utilizes all cluster information acquired in the training phase in order to minimize the impact of noise on the prediction efficiency, .

1.4. Article Structure

This article is organized as follows: Sec. 2 introduces the key-concepts on which HyCASTLE is built upon, Sec. 3 shows in details the HyCASTLE architecture and its train and prediction phases and includes the complexity analysis. Sec. 4 presents the analysis of the parameters, the experiments results and the corresponding analysis, while the conclusions are drawn in Sec. 5.

2. Theoretical Premises

Introduced by Angelov et al. [24], the *typicality* is a non-parametric estimator for discrete data sets, derived using only statistical properties of the data themselves. In connected graph theory, the *centrality* [26] is defined as the average length of the shortest path between a node in the graph and all other nodes. Given an N-dimensional data set, for each instance x_i, we can compute the *cumulative proximity* $q_N(x_i)$, which can be recognised as an inverse centrality with a squared distance:

$$q_N(x_i) = \sum_{j=1}^{N} d^2(x_i; x_j) \tag{1}$$

where $d^2(x_i; x_j)$ is the squared distance between the instance x_i and every other instance x_j. From the cumulative proximity, we can define the *Eccentricity* $\epsilon_N(x_i)$ that quantifies data samples away from the mode:

$$\epsilon_N(x_i) = \frac{2q_N(x_i)}{\frac{1}{N}\sum_{j=1}^{N} q_N(x_j)}. \tag{2}$$

The *Discrete local density* $D_N(x_i)$ is the inverse of the Eccentricity, and the typicality $\tau_N(x_i)$ is the normalized Discrete local density:

$$\tau_N(x_i) = \frac{D_N(x_i)}{\sum_{j=1}^{N} D_N(x_j)}. \tag{3}$$

The typicality resembles an uni-modal probability mass function and, when combined with the *Chebyshev Inequality* [27]:

$$|\tau(x_i) - \mu_\tau| > 3\sigma_\tau \tag{4}$$

where $\tau(x_i)$ is the typicality of the scrutinized instance, μ_τ is the average data typicality and σ_τ its standard deviation, it can be used for outliers detection. Angelov et al. [24] showed that typicality possesses all the properties of a Probability Density Function (PDF) and thus can be used to capture the topological structure of the data. In particular, it can be used to capture local modes in the data and form shape-free clusters around the local maxima of the typicality distribution.
Our proposed method builds upon the work developed by Angelov et al. [24] and uses the typicality clusters as the pre-processing step of a classification algorithm.

The Gini Index or Coefficient [28] is a measure of statistical dispersion that quantifies the inequalities among values of a frequency distribution. A Gini coefficient of 1 expresses the maximum inequality among values. Given a set of instances with J classes with $i \in 1, 2, ..., J$ and p_i the fraction of items labeled with class i in the set, we define the Gini Index as:

$$I_G(p) = 1 - \sum_{i=1}^{J} p_i^2. \tag{5}$$

3. HyCASTLE

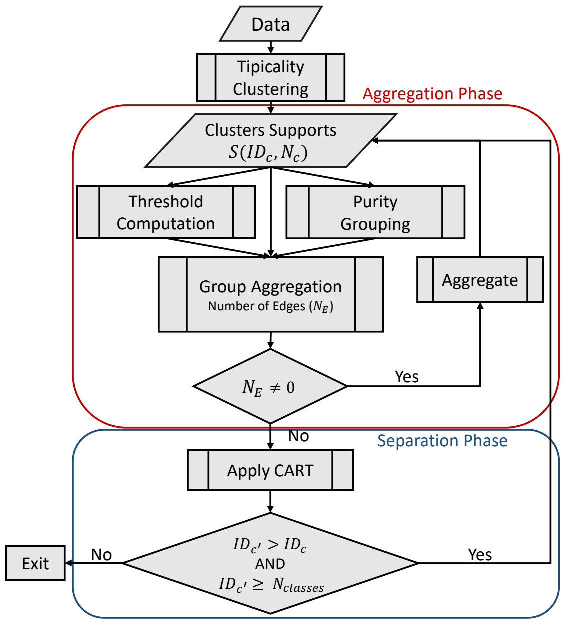

Figure 1: HyCASTLE training workflow. The two coloured blocks represent, respectively, the Aggregation phase (red) and the Separation Phase of the algorithm training (blue). For a detailed explanation of the flowchart refer to Section 3.1.

HyCASTLE is an iterative supervised classification algorithm aiming at finding a cluster configuration that maximizes cluster purity (i.e. the percentage of data correctly assigned to the right cluster). Its workflow is shown in Figure 1.
We define a cluster as a set of input instances that form a high typicality connected region, which is separated from other clusters by low typicality connected regions. The proposed method relies on geometrical and topological features (a similarity measure and the inherent structure of the parameter space derived from the typicality distribution) to find and aggregate clusters while using the Gini index to measure the wrong assignments to the clusters and to modify the cluster configuration in order to improve the classification task. The proposed method requires as parameters a *metric* in order to compute distances both between instances and clusters, a distance threshold α necessary to select clusters that are close enough in the parameter space to be selected as candidates for aggregation and k, i.e. the number of points of each cluster that are selected as representatives of that cluster in prediction. Usually, both graph and density-based clustering methods require a parameter that is used as a threshold to decide if two instances or cluster are connected. While this parameter usually has a constant impact in training [23, 29, 30], in HyCASTLE case, since cluster shapes and support sizes are changing during training, their mutual distances are changing too and thus the threshold α parameter has a dynamic impact.

3.1. HyCASTLE Training

As in many others distance-based algorithms such as the k-nearest neighbour classifier [31] or k-means [32], choosing the most appropriate distance metric becomes crucial to solve a specific classification problem. Usually, in real-world datasets, features ranges may arbitrarily vary: some features could present variations of order of magnitude higher than others, and some metrics, like the *Euclidean*, may suffer heavily from such disparity of variation ranges, impacting on their final performance.
Therefore, a mandatory pre-processing step of the proposed algorithm consists of normalizing the numerical attributes by modifying their values through the following scaling formula:

$$x_i = (x_i - min(X))/(max(X) - min(X)) \tag{6}$$

where x_i is an attribute value for the current instance and X is the entire range of values of that attribute. Moreover, because HyCASTLE cannot handle categorical attributes if an ordering could be established, we employed an Ordinal Encoding [1]; otherwise, we employed a One-Hot encoding, also known as 1-of-K scheme [1], to convert the categorical attributes in numerical ones.

The algorithms then proceeds (*Typicality Clustering* in Figure 1) by computing the typicality distribution $\tau_N(X)$ of the data through Eq. 3.
Outliers may affect the typicality distribution and, thus, they must be identified and excluded from the rest of the procedure. We use the PDF-like properties of the typicality distribution and combine them with the Chebyshev Inequality through Eq. 4. If an instance satisfies Eq. 4, then it is labelled as an outlier and removed from the data.

After outlier removal, the typicality distribution $\tau_N(X)$ must be recomputed through Eq. 3. To define the local modes in typicality, data are ordered starting with the instance presenting the highest typicality value. Then the local maxima in typicality are found by a simple comparison of neighbouring values and, through the geometrical information (distance), all instances are assigned to these local maxima (or pivots) to form the initial set of clusters. All cluster characteristics, i.e. their index ID_c and support size N_c, are stored in a dictionary $S(ID_c, N_c)$ and each instance membership is defined through the ID_c of the closest pivot. The pseudo-code of the Typicality Clustering phase is shown in the Algorithm 1.

Given that the typicality-based clustering reflects the actual data distribution, it may not be optimal to deal with a classification problem. This eventuality may come from the existence of clusters sharing the same labels and close enough to be unified or due to the necessity of separating clusters containing instances with different labels if these instances are geometrically separable. The underlying hypothesis is that the optimal cluster configuration lays between the data-driven clustering found through typicality and simple label-driven cluster separation. Thus it begins the *aggregation phase* of the algorithm (red box in Figure 1); this phase is composed of the three steps that are hereafter listed:

1. *Threshold Computation*: all cluster centroids, i.e. the clusters barycenters, are computed and their mutual distances distribution is recorded. Then the potential neighbours threshold is computed as follows:

$$t_d = \alpha \cdot mean(D_c) \tag{7}$$

   where α is a user defined parameter and D_c is the cluster's centroids distance distribution. The pseudo-code of this step is shown in the Algorithm 2;
2. *Purity Grouping*: clusters are separated on the basis of their member's labels in a number of groups equaling the number of detected classes in the data plus one containing spurious clusters. This operation is performed in order to decrease the computational time required to perform the aggregation given that for pure groups no Gini index must be computed in order to aggregate clusters. The pseudo-code of this step is shown in the Algorithm 3;
3. *Group Aggregation*: a graph is constructed from each group of clusters found in the *purity grouping* step. Each node of the graph represents a cluster and is defined through the cluster centroid

Algorithm 1 Typicality Clustering

```
Require: data $D$, metric, labels $L$
Ensure: data_ranked $D$, labels_ranked $L$, supports idxs $S$, assignments $A$, outliers $O$
 1: $DR \leftarrow \emptyset$
 2: $P \leftarrow \emptyset$
 3: $\mathcal{S} \leftarrow \emptyset$
 4: for each $x_i \in D$ do
 5:     $\tau(x_i) \leftarrow$ compute typicality of $\tau(x_i, TD)$ with equation 3
 6: end for
 7: $\mu_\tau \leftarrow mean(\tau)$
 8: $\sigma_\tau \leftarrow std(\tau)$
 9: for every $x_i \in D$ do                ▷ Outlier detection phase
10:     if $x_i$ satisfyes Eq. 4 then
11:         $D \leftarrow D \setminus \{x_i\}$
12:     end if
13: end for
14: for every $x_i \in D$ do
15:     $\tau(x_i) \leftarrow$ compute typicality of $\tau(x_i, TD)$ with equation 3
16: end for
17: $x \leftarrow reorder\_vector(\tau)$
18: $x_M \leftarrow max(\tau)$
19: $D \leftarrow D \setminus \{x_M\}$
20: $DR \leftarrow DR \cup \{x_M\}$
21: while D $\neq \emptyset$ do
22:     $x_i \leftarrow$ find the closest point in P to $x_M$
23:     $D \leftarrow D \setminus \{x_i\}$
24:     $DR \leftarrow DR \cup \{x_i\}$
25: end while
26: $P \leftarrow P \cup \{x_M\}$
27: for every $x_i$ in DR do
28:     if $\tau_{x_{i-1}} < \tau_{x_i} \wedge \tau_{x_{i+1}} < \tau_{x_i}$ then
29:         $P \leftarrow P \cup \{x_i\}$
30:     end if
31: end for
32: if $\tau_{x_N} > \tau_{x_{N-1}}$ then
33:     $P \leftarrow P \cup \{x_N\}$
34: end if
35: $\mathcal{S} \leftarrow \{S_0, \ldots, S_{len(P)}\}$
```

and the cluster Gini index. For each pure group, all distances between nodes are scrutinized and an edge is added between two nodes if their mutual distance is smaller then the potential neighbour threshold. For the spurious group, it is necessary to compute also the Gini index that would result from the aggregation of any couple of clusters that pass the distance-based check. Thus with the parameter α, the user is choosing the level of topological constraining (unknown information extracted from the data) to apply on a label-based aggregation criterium. After all edges are formed, all found sets of connected components couples are aggregated in parallel and the training data assignments indexes are updated. The pseudo-code of this step is shown in the Algorithm 4;

The three steps loop continues until no cluster couples satisfy the distance-based threshold (t_d) or when the number of clusters equals the number of detected classes in the data. The pseudo-code of the full aggregation phase is shown in the Algorithm 5.

Algorithm 2 Threshold Computation

```
Require: data $D$, assignments $A$, supports idxs $S$, metric, $\alpha$
Ensure: centroid distance distribution $D_c$, threshold $t_d$
 1: $(ID_c, N_c) \leftarrow S$
 2: $B_c \leftarrow \emptyset$
 3: $D_c \leftarrow \emptyset$
 4: for every $id$ in $ID_c$ do
 5:     $B_c \leftarrow sum(D[A = id])/N_c$
 6: end for
 7: $D_c \leftarrow$ distances between all centroids $B_c$
 8: compute threshold $t_d$ through Eq. 7
```

Algorithm 3 Purity Grouping

```
Require: assignments $A$, supports idxs $S$, labels $L$
Ensure: separated supports idxs $S_G$
 1: $(ID_c, N_c) \leftarrow S$
 2: classes $C \leftarrow$ unique elements in $L$
 3: $S_G \leftarrow \emptyset : len(S_G) = len(C) + 1$
 4: for every $id$ in $ID_c$ do
 5:     $L_c \leftarrow L[A = id]$
 6:     if there is only one unique element $u$ in $L_c$ then
 7:         $S_G[c = u] \leftarrow ID_c$
 8:     else
 9:         $S_G[-1] \leftarrow ID_c$
10:     end if
11: end for
```

Algorithm 4 Group Aggregation

```
Require: separated supports idxs $S_G$, labels $L$, assignments $A$, centroid distance distribution $D_c$, threshold $t_d$
Ensure: supports idxs $S$, assignments $A$
 1: for every group $s_G$ in $S_G$ do
 2:     Create graph $G(ID_c, Gini(L_c)$ using as attributes all indexes and Gini indexes of clusters $c$ in the group $s_G$
 3:     if $s_G$ is pure then
 4:         for every couple of nodes $G_ij$ in $G$ without repetition do
 5:             if $D_c[i, j] \leq t_d$ then
 6:                 add an edge $E_ij$ between the nodes $i$ and $j$
 7:             end if
 8:         end for
 9:         find all the sets of connected components in $E_ij$ and aggregate them
10:         merge $ID_c$ of all aggregated connected components and update $N_c$.
11:     else
12:         for every couple of nodes $G_ij$ in $G$ without repetition do
13:             if $D_c[i, j] \leq t_d$ and $Gini(L_c[i]) + Gini(L_c[j]) <= Gini(L_c[i] \cup L_c[j])$ then
14:                 add an edge $E_ij$ between the nodes $i$ and $j$.
15:             end if
16:         end for
17:         find all the sets of connected components in $E_ij$ and aggregate them
18:         merge $ID_c$ of all aggregated connected components and update $N_c$.
19:     end if
20: end for
21: $S = S \cup S_G$
22: update $A$ with the updated $ID_c$
```

Algorithm 5 Aggregation Phase

```
Require: data $D$, labels $L$, supports idxs $S$, assignments $A$, metric, $\alpha$
Ensure: supports idxs $S$, assignments $A$
  classes $c \leftarrow$ unique elements in $L$
  while $len(S) \geq len(c)$ do
      $D_c, t_d \leftarrow$ threshold computation(D, A, S, metric, $\alpha$)
      $S_G \leftarrow$ purity grouping(A, S, L)
      $AS \leftarrow$ group aggregation($S_g$, L, A, $D_c$, $t_d$)
  end while
```

As soon as the *aggregation phase* terminates, it begins the *separation phase*. This phase aims at splitting each cluster into sub-clusters, using both geometrical and problem-specific information. i.e. instance labels are used to estimate the purity of each cluster and split it in order to get the as pure sub-clusters as geometrically possible. This phase is necessary in order to exit from the local minima of the Gini Index, coming from both the initial typicality based clustering and the aggregation phase. Given the local nature of both aggregation and separation, this process has the effect of removing instances from each cluster that has a weak geometrical connection with the cluster (in other words, instances which are in the outskirt of the cluster) and/or in order to minimize the Gini Index, i.e. separating instances with different labels. A CART is trained on each cluster and then is asked to return the leaf index that predicted each cluster instance. The original cluster is then replaced by as many clusters as the number of leaves in the tree. If the clusters configuration after the separation phase contains more clusters than in the previous step, i.e. if any of the trees had more than one terminal node, then the algorithm goes back to the beginning of the aggregation phase, and the cycle starts again. This cycle, as we demonstrate in Sec. 4.4 by injecting noise in the data, is inherently robust to class noise because an instance topologically similar to the rest of the cluster, even if mislabeled, would be still kept inside the cluster and not separated and, in the same way, an instance sharing the same label as a given cluster but to topologically different from it would be still separated from the cluster. The algorithm stops if no cluster separation is registered or if the previous cycle cluster configuration after separation is met again.

Algorithm 6 Separation Phase

```
Require: data $D$, labels $L$, supports idxs $S$, assignments $A$
Ensure: Supports idxs $S$, assignments $A$
 1: for every cluster $C$ do
 2:     if $len(C) > 1$ then
 3:         $l \leftarrow$ labels in Cluster
 4:         $DT \leftarrow$ CART model
 5:         DT.fit on $(C, l)$
 6:         $LA \leftarrow \emptyset$
 7:         for every $x_k$ in Cluster do
 8:             $LA \leftarrow$ leaf assignments $la_k$
 9:         end for
10:         if len(LA) > 1 then
11:             $S \leftarrow LA$
12:             for every $x_k$ in Cluster do
13:                 $A_k \leftarrow la_k$
14:             end for
15:         end if
16:     end if
17: end for
```

3.2. HyCASTLE Prediction

In the prediction phase, the algorithm computes the probability that a given instance may belong to all cluster present in the cluster configuration found in training by computing its mean distance from the k closest members of each cluster. The mean distance distribution is converted into a probability distribution through the following equation:

$$P_d(c_i) = \frac{(1/d_{c_i})}{\sum_{i=1}^{M}(1/d_{c_i})} \tag{8}$$

where $P_d(c_i)$ is the distance-based probability that the instance belongs to the cluster c_i and d_{c_i} is the mean distance between the instance and the closest k members of the cluster. To output the class probability, all probabilities of single-class clusters, i.e. pure clusters, are summed to form the probability of their related classes while probabilities relative to multi-class clusters, i.e. spurious clusters, are divided among class probabilities in proportion to each class frequency found in the cluster. Given the explained probabilistic nature of the prediction, we expect it to be resilient to class noise present in the data.

3.3. Complexity Analysis

Assume that dataset D contains N samples with F attributes, the typicality clustering phase requires: *(i)* the computation of all distances between the N samples to generate the typicality distribution and *(ii)* the instances ordering HyCASTLEting with the highest typicality object to find local maxima of the typicality distribution around which the clusters are generated. In the worst possible scenario, such operations present a computational complexity of $O((FN)^2)$. In the aggregation and separation phases, given that the number of found clusters through typicality is $M < N$, the computational complexity goes respectively as $O((FM)^2)$ in aggregation and as $O(MlogN)$ in separation. Thus the total computational complexity of the proposed method is $O((FN)^2)$.

4. Experiments

In this section, we describe the dataset used for evaluating the proposed model performance, we analyze the effect of parameter choices on classification performance, we compare the proposed model with the AbDG model presented in Bertini et al. [12], the two models presented in Xiao et al. [7] HCFC-K and HCFC-D, the K-means+ID3 model proposed in Gaddam et al. [9], the TBWC model proposed in Kaewchinport et al. [6], the RK+SVM model proposed by Rajomahamed et al. [8] and six classical supervised classificators C4.5 [14], KNN [16], SVM[17], NB [18], LR [19] and MLP [20] and then we compare the classification performances of HyCASTLE, HCFC-K, HCFC-D, and other supervised classifcators on a dataset containing class noise [33]. To compare HyCASTLE performance with both the AbDG [12] model and the HCFC-K and HCFC-D [7] models we employed the same experimental strategies used in the respective articles and compared our results with the best results reported in the respective papers. In order to compare with Bertini et al. [12], we employed a ten times repeated double 10-fold stratified cross-validation strategy in which the inner loop selects the best model parameters using the validation set, while the generalization errors are estimated using a previously unseen test set. In order to compare with Xiao et al. [7], we employed a ten times repeated fivefold-cross-validation strategy. First, each dataset was divided in five random subsets. Then iteratively one set was selected as the test set, one was selected from the remaining four set as the validation set and the remaining three as the training set. Each model was trained on the training set, its parameters were optimized on the validation set through a grid-search, and its performance was evaluated on the test set. The process was then repeated five times to ensure that each of the subsets was used as test set once. The reported model performance on a dataset is then the average performance on the five folds. The grid-search intervals of the remaining models were selected following the guidelines provided by their authors in the respective papers or, if not provided, given our knowledge about the models:

- both K-means+ID3 and RK+SVM need to optimize the number of clusters K and thus following the prescription in [9] and [8] K was respectively searched in the interval $[2, 20]$ and in the interval $[2, 10]$ with a step-size of 1;
- TBWC needs to optimize the number of trees (n) and the optimal number of clusters K. Following the prescription in [6], we searched for the optimal n and K in the interval $[5, 40]$ with a step-size of 5;
- KNN needs to optimize the number of clusters K and thus K was searched for the optimal value in the interval $[1, 20]$ with a step-size of 1;
- MLP was initialized with two hidden layers with a number of neurons equaling to $2 \cdot n - 1$ in the first layer and $n + 1$ in the second layer where n is the number of input features and a ReLU activation function;
- SVM was implemented through the Scikit-learn library. [34], we selected a radial basis function kernel and, as advised by the authors of the library, the kernel parameter γ was selected as $1/n$ with n the number of input features;
- C4.5 was implemented through the Chefboost library. [35].

HyCASTLE, as discussed in Sec. 3, needs to optimize 3 parameters to find the optimal solution: *(i)* the *metric* used for distance calculation was selected among the *Euclidean*, *Cityblock*, *Cosine* and *Minkowski* distances; *(ii)* the α value, used to control the topology-based selection of candidates for aggregation, was selected in the interval $[1, 20]$ with an increment of 1 and *(iii)* the k value, used to select the number of representative points for each cluster in prediction, was selected in the interval $[1, 30]$ with an increment of 1. Because NB and K-means-ID3 can process only discrete attributes, we made use of the procedure described in Fayyad and Irani. [36] to discretize any continuous attribute before applying the models and because HyCASTLE cannot handle categorical attributes if an ordering could be established, we employed an Ordinal Encoding [1]; otherwise, we employed a One-Hot encoding, also known as 1-of-K scheme [1], to convert the categorical attributes in numerical ones.

4.1. Parameter Analysis

In this subsection, we analyze the effects of parameter choices on HyCASTLE classification performance. Specifically, we fixed the *metric* as the *Euclidean* distance for simplicity and considered the effect on the performance of the α coefficient and of the number of cluster representatives k only. We decided not to analyze the effect of the choice of *metric* on the model performance for two main reasons: *i*) the best metric is completely dependent upon the data and its effect are not easily explained and *ii*) it would have made the effect of the other two parameters much harder to explain. Nevertheless by comparing the best accuracy shown in Fig. 2 and 3 with the ones relative to the same datasets in Table 2 and 4, it can be seen that, as stated in Sec. 3, the choise

Dataset	Instance	Attribute	Class
Abalone	4177	8	28
Auto	205	26	6
Balance	625	4	3
Blood-Transfusion (BT)	748	5	2
Connectionist	208	60	2
Credit Approval	690	15	2
Dermatology	366	33	6
E.coli	336	8	8
Flags	194	30	8
Flare	1389	10	2
Frogs-MFCCs-Family (FMF)	7196	22	4
Glass	214	10	6
Haberman	306	3	2
Image Segmentation	2310	19	7
Iris	150	4	3
Magic	19020	11	2
Monk2	432	7	2
Parkinsons	197	23	2
Pima	768	8	2
Post Operative	90	8	3
Seeds	210	7	3
Soybean	307	35	15
Sonar	208	60	2
Teaching-Assistant-Evaluation (TAE)	151	5	3
User-Knowledge-Modeling (UKM)	403	5	4
Vertebral-column-2C (VC2)	310	6	2
Vertebral-column-3C (VC3)	310	6	3
Waveform	5000	21	3
Wholesale Customer	440	8	2
Wilt	4889	6	2
Wine	178	13	3
Breast Cancer Wisconsin Diagnostic (WDBC)	569	30	2
Breast Cancer Wisconsin Prognostic (WPBC)	198	34	2
Yeast	1484	8	9
Zoo	101	17	7

Table 1: UCI Datasets. [25] for Classification Experiments; in order the dataset name, the number of instances, the number of attributes and the number of classes.

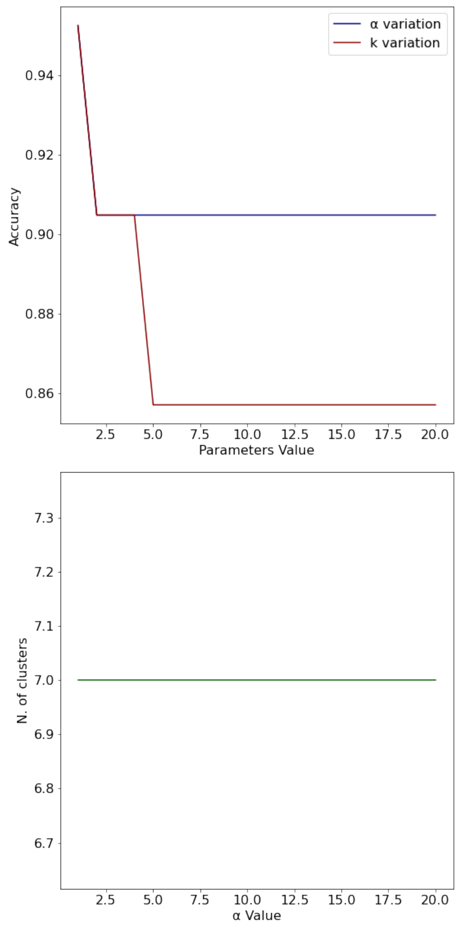

Figure 2: On the top panel, it is shown the classification accuracy obtained on the Zoo dataset from UCI. [25] by constant variation of the α (blue plot) and k parameters value in the range $[1, 20]$; on the lower panel, the number of clusters in the final cluster configuration obtained through α variation.

of *metric* has a quite consistent effect on HyCASTLE performance. All tests were performed by splitting each dataset in train, validation, and test sets with the following percentages $[0.6, 0.2, 0.2]$. Given the lack of prior knowledge of the effect of one parameter over the other, we fixed the variation of both parameters in the interval $[1, 20]$ and then, supposing we wanted to measure the effect of the first parameter, we firstly fixed the value of both parameters to 1 and trained the model on the train set. Then we searched for the optimal value of the second parameter on the validation set. Having found it, we fixed it and iteratively trained and evaluated the model by increasing the value of the first parameter each time. The same thing was performed by switching the first parameter with the second. Given the α also control the granularity of the produced clustering, we also recorded the number of final clusters for each value of α. The analysis, whose results are summarized in Fig. 2 and 3,

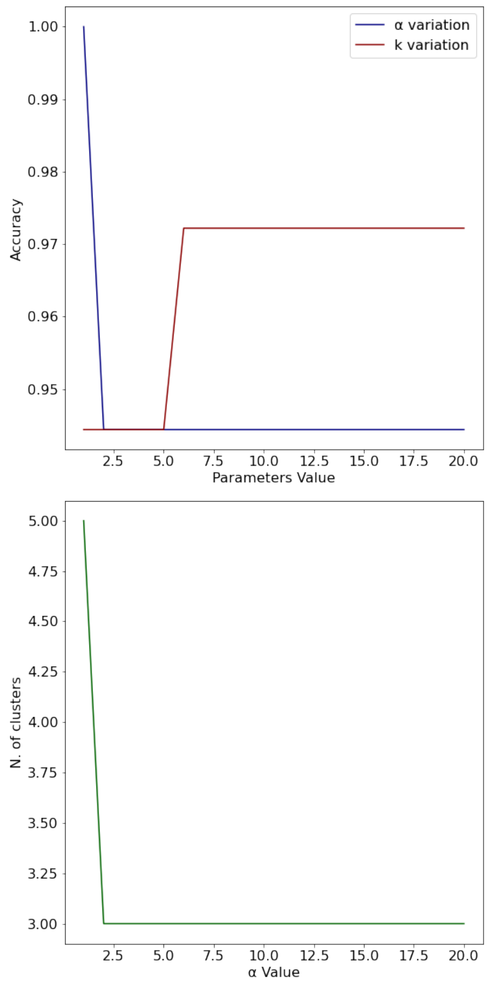

Figure 3: On the top panel, it is shown the classification accuracy obtained on the Wine dataset from UCI. [25] by constant variation of the α (blue plot) and k parameters value in the range $[1, 20]$; on the lower panel, the number of clusters in the final cluster configuration obtained through α variation.

was carried on two exemplary datasets from the ones listed in Table 1 Wine and Zoo. As it is seen the number of clusters, as explained in Sec.3.1 has the number of classes detected in the training data as a lower bound (respectively 7 for Zoo and 3 for Wine) and its trend is completely dictated by the value of α. In Wine, see Fig 3, as α increases, the number of clusters that pass the distance-based threshold increase and thus more clusters are aggregated with the results of having the number of clusters in the final cluster configuration dropping almost immediately to 3. At the same time, the model accuracy decreases, showing how having 5 clusters in the final cluster configuration with two classes each split over 2 sub-clusters helps in the prediction phase. In this case the model shows a better accuracy with a value of K of at least 5. In Zoo, we find a similar situation for the α value, the number of clusters it is always equal to the number of classes while the accuracy decreases with an increasing k value. It is worth noticing that when α reaches a high enough value, then its effect on the model performance stabilizes. This is explained by the fact that when α reaches a given dataset dependent value, all clusters pass the distance-based threshold and thus any further increment of α does not impact in any meaningful way.

4.2. *Evaluating HyCASTLE Performance*

Table 2 shows the classification accuracy of HyCASTLE compared with the other 11 models on 23 of the 35 datasets listed in Table 1. These datasets were selected in order to directly compare with Xiao et al. [7]. As it can be seen, HyCASTLE shows a higher mean classification accuracy with respect to the other models. In order to test if statistical differences between the 12 models classification performances can be considered significant, we employed the Friedman test [37] and the Iman-Davenport test [38] with a null hypothesis: "all models show the same classification performance". The statistic of the Friedman test is distributed following and F-distribution. The degrees of freedom of the F-distribution are given by $(N_a - 1)$ and $(N_a - 1) \times (N_d - 1)$ where N_a is the number of tested algorithms and N_d is the number of dataset used for the comparison. In this case $N_a = 12$ and $N_d = 23$ and thus the critical value for a confidence level of $\alpha = 0.05$ is $F(11, 242) = 1.83$. Any value higher than the critical value guarantees the rejection of the null hypothesis with a confidence value of 95%. Additionally we performed a Nemenyi *post-hoc test* [39] with a significance level of $\alpha = 0.05$. The test is based on the absolute difference of the average rankings of the classifiers. Given the significance level, the test determines the *critical difference* ($CD = 3.50$ in

Dataset	HyCASTLE	HCFC-K	HCFC-D	C4.5	SVM	LR	KNN	NB	MLP	RK+SVM	K-means+ID3	TBWC
Abalone	**28.12 ± 0.12**	26.05 ± 0.46	25.91 ± 0.68	23.15 ± 1.92	25.12 ± 2.46	23.78 ± 1.94	25.95 ± 0.67	24.53 ± 0.74	26.88 ± 0.39	24.23 ± 2.16	23.01 ± 1.15	24.92 ± 1.35
Auto	73.12 ± 2.28	**76.10 ± 2.02**	74.39 ± 3.19	74.09 ± 2.97	67.14 ± 5.87	63.89 ± 5.48	68.12 ± 8.71	45.32 ± 7.83	69.38 ± 7.5	73.28 ± 7.89	70.64 ± 4.12	61.49 ± 5.12
BT	**80.00 ± 2.06**	78.48 ± 3.54	77.94 ± 3.24	76.94 ± 4.03	76.22 ± 3.98	76.92 ± 3.64	79.6 ± 1.61	71.12 ± 3.57	76.27 ± 1.37	69.87 ± 4.76	75.16 ± 3.89	74.23 ± 4.43
Connectionist	**86.19 ± 3.16**	79.88 ± 5.35	76.68 ± 6.92	72.56 ± 6.17	74.39 ± 6.28	75.94 ± 4.46	81.43 ± 6.63	64.25 ± 5.98	78.1 ± 5.08	70.01 ± 9.66	76.98 ± 3.12	73.96 ± 3.21
Dermatology	99.16 ± 1.66	93.98 ± 2.90	94.27 ± 3.60	91.00 ± 3.25	97.25 ± 1.96	98.16 ± 1.84	83.61 ± 2.39	84.25 ± 3.45	**99.44 ± 1.11**	91.15 ± 1.93	88.17 ± 5.64	79.22 ± 4.34
Ecoli	**88.35 ± 3.70**	83.05 ± 4.50	83.78 ± 2.75	79.18 ± 3.13	81.78 ± 4.15	81.16 ± 4.84	88.06 ± 3.40	72.60 ± 4.12	55.59 ± 10.58	80.98 ± 3.99	79.99 ± 4.91	81.48 ± 3.0
Flags	**69.47 ± 2.05**	63.43 ± 3.07	63.66 ± 2.53	62.00 ± 3.00	34.15 ± 3.97	48.23 ± 8.11	67.89 ± 6.94	39.81 ± 5.25	63.08 ± 6.61	34.21 ± 1.83	61.12 ± 2.14	44.81 ± 4.75
Flare	**87.15 ± 1.98**	84.70 ± 4.25	85.11 ± 3.00	81.76 ± 8.84	84.16 ± 3.49	84.48 ± 4.29	86.76 ± 1.66	56.38 ± 5.74	86.17 ± 0.76	83.77 ± 3.44	83.12 ± 5.98	84.12 ± 4.33
FMF	95.46 ± 1.83	96.13 ± 0.52	95.57 ± 0.82	93.29 ± 1.14	92.72 ± 0.84	92.12 ± 0.65	70.76 ± 0.23	64.29 ± 1.63	**98.37 ± 0.24**	93.74 ± 0.87	93.55 ± 0.94	93.79 ± 1.98
Glass	**74.42 ± 3.28**	70.73 ± 3.55	70.78 ± 8.64	62.44 ± 8.12	48.13 ± 4.67	61.84 ± 3.27	70.71 ± 3.48	59.46 ± 7.27	68.84 ± 4.79	62.04 ± 7.73	67.21 ± 3.76	58.21 ± 7.31
Magic	84.36 ± 2.26	82.51 ± 0.49	80.81 ± 0.91	80.15 ± 0.83	67.82 ± 0.68	77.65 ± 2.16	83.61 ± 0.38	71.36 ± 1.48	**86.36 ± 0.65**	68.32 ± 1.55	80.87 ± 0.44	78.99 ± 0.65
Monk2	**90.83 ± 1.97**	86.36 ± 2.87	86.20 ± 3.10	79.22 ± 2.12	65.43 ± 1.89	68.14 ± 1.46	73.67 ± 2.27	59.97 ± 3.64	67.93 ± 2.12	63.39 ± 1.48	80.21 ± 7.13	66.13 ± 3.19
Parkinsons	**94.36 ± 1.32**	91.28 ± 3.49	87.95 ± 6.13	81.98 ± 4.96	72.88 ± 4.87	85.00 ± 1.92	93.33 ± 2.61	74.39 ± 9.08	89.74 ± 4.29	75.87 ± 3.12	86.17 ± 3.41	80.96 ± 4.22
Seeds	**93.39 ± 2.10**	91.90 ± 2.72	90.48 ± 5.39	91.02 ± 5.12	93.27 ± 2.16	91.78 ± 2.42	92.86 ± 3.98	88.45 ± 5.12	92.86 ± 3.98	93.72 ± 2.94	90.77 ± 7.98	90.14 ± 6.21
Soybean	81.51 ± 3.36	89.67 ± 3.29	88.96 ± 2.05	84.76 ± 0.98	**91.26 ± 4.64**	85.14 ± 2.45	83.77 ± 1.51	83.97 ± 3.76	88.52 ± 4.6	86.41 ± 4.55	84.19 ± 4.12	71.54 ± 2.46
TAE	**66.66 ± 5.96**	64.90 ± 3.85	66.55 ± 4.94	60.28 ± 4.12	44.12 ± 2.99	46.13 ± 2.67	59.33 ± 5.73	42.87 ± 5.91	49.68 ± 8.06	51.95 ± 2.12	59.38 ± 5.14	44.63 ± 3.43
UKM	91.83 ± 0.78	92.31 ± 0.58	91.07 ± 0.37	89.47 ± 0.76	88.64 ± 5.78	72.36 ± 7.61	88.64 ± 4.38	88.56 ± 4.93	**96.54 ± 1.44**	91.13 ± 7.11	88.13 ± 7.45	91.78 ± 2.72
VC2	**84.51 ± 2.15**	83.55 ± 2.45	82.58 ± 4.90	81.48 ± 3.81	74.35 ± 3.64	79.96 ± 5.32	81.61 ± 4.96	67.15 ± 2.96	83.55 ± 2.77	75.89 ± 6.77	80.93 ± 1.94	80.66 ± 6.0
VC3	81.61 ± 4.27	81.29 ± 2.65	**83.39 ± 2.91**	78.12 ± 4.29	79.96 ± 4.65	82.16 ± 4.58	80.97 ± 3.29	71.69 ± 4.13	75.81 ± 13.91	74.19 ± 9.07	78.16 ± 2.69	80.12 ± 4.17
WC	**93.86 ± 3.09**	90.68 ± 2.67	90.16 ± 1.23	87.98 ± 2.37	71.35 ± 6.14	90.01 ± 3.92	92.5 ± 2.93	69.16 ± 2.98	67.05 ± 18.62	88.13 ± 5.38	85.63 ± 1.83	75.01 ± 1.84
Wilt	97.04 ± 0.31	**98.22 ± 0.18**	97.59 ± 0.43	95.96 ± 0.98	95.14 ± 1.62	96.03 ± 0.72	95.68 ± 0.62	88.47 ± 0.65	94.63 ± 0.0	94.63 ± 0.43	94.99 ± 4.55	94.99 ± 0.93
Yeast	**93.86 ± 3.09**	51.49 ± 2.84	53.40 ± 2.63	44.39 ± 4.12	57.49 ± 3.60	50.61 ± 2.67	58.72 ± 2.22	26.12 ± 3.41	59.19 ± 2.01	50.06 ± 2.66	55.83 ± 4.97	51.83 ± 2.15
Zoo	97.00 ± 2.00	**99.05 ± 2.64**	97.55 ± 2.73	91.10 ± 2.53	94.89 ± 8.23	88.04 ± 6.83	96.0 ± 5.83	91.25 ± 4.93	94.29 ± 4.67	88.39 ± 7.80	96.46 ± 3.21	93.58 ± 7.24
Average	**84.14 ± 3.69**	80.68 ± 2.65	80.22 ± 3.18	75.28 ± 3.46	72.94 ± 3.85	74.76 ± 3.62	78.41 ± 3.21	65.45 ± 4.29	76.88 ± 4.59	73.28 ± 4.31	77.42 ± 3.94	72.9 ± 3.7
Percentage	60.87%	13.04%	4.35%	0.00%	4.35%	0.00%	0.00%	0.00%	17.39%	0.00%	0.00%	0.00%

Table 2: Comparison of Classification Accuracy (Mean ± Standard Deviation %) between HyCASTLE, the two models presented in Xiao et al. [7] HCFC-K and HCFC-D, C4.5 [14], SVM[17], LR [19], KNN [16], NB [18], MLP [20], the RK+SVM model poposed by Rajomahamed et al. [8], the K-means+ID3 model proposed in Gaddam et al. [9] and the TBWC model proposed in Kaewchinport et al. [6]. The last two rows show respectively the average score over all datasets and the winning percentage for each model. In bold are highlighted the best model for each dataset.

our case) and if the difference between the average rankings of two given algorithms is greater then the CD, then the null hypothesis that the two algorithms show the same classification performance is rejected. Both the

Method	Test Value	Critical Value	Hypothesis
Friedman	124.37	19.67	rejected
Iman-Davenport	21.27	1.83	rejected

Table 3: Results of the Friedman and Iman-Davenport tests for testing the statistical differences between the 12 models performances shown in Table 2.

Friedman and Iman-Devenport test statistics followed a χ^2 distribution with 11 degrees of freedom and, as it is seen in Table 3 both tests values exceed their critical values. Thus we can reject the null hypothesis, i.e. at a 95% confidence level the model's classification performances show differences that can be deemed as statistically significant.

Figures 4 and 5 show the Nemenyi *post-hoc test* results. In the first figure, if models are connected with a line, it means that no significant difference in performance can be detected. In the second, each node is ranked by its model CD value, the lower it is, the more different is the performance of that model from the rest. If the absolute difference between the CD values of two nodes is smaller then critical difference, then the nodes are connected and no significant difference in performance can be detected between the two relative models. From both figures, it can be deduced that no statistically significant difference is shown at a 95% interval between NB, SVM, C4.5, LR, TBWC, RK+SVM

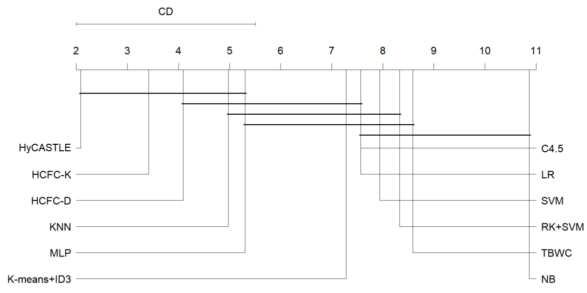

Figure 4: Critical difference plot:any two algorithms not joined by a line may be regarded as having different classification performances. The plot refers to the results shown in Table 2.

and K-means+ID3. HCFC-D and HCFC-K show better performance when compared to the other models, and their results are compatible with KNN, MLP and K-means+ID3. HyCASTLE, nevertheless has a compatible performance with both HCFC-D and HCFC-K and KNN, shows to have a performance significantly different from all other models and a lower CD when compared to the other hybrid models. Furthermore from the statistics presented as box plots in Fig. 6, it can be seen that HyCASTLE, besides having the highest accuracy (median value in highlighted in red), has the lowest variation and thus is performance is more consistent over the experiments when compared to the performance of the other models.

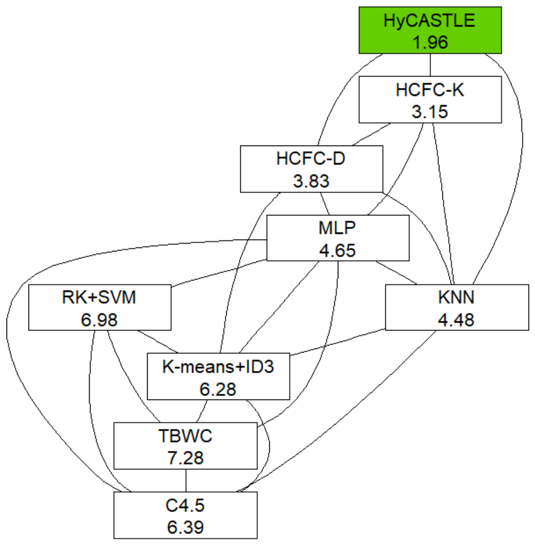

Figure 5: Detailed Critical difference plot: if two nodes are joined it means that the null hypothesis of having different performances cannot be rejected. The plot refers to the results shown in Table 4.

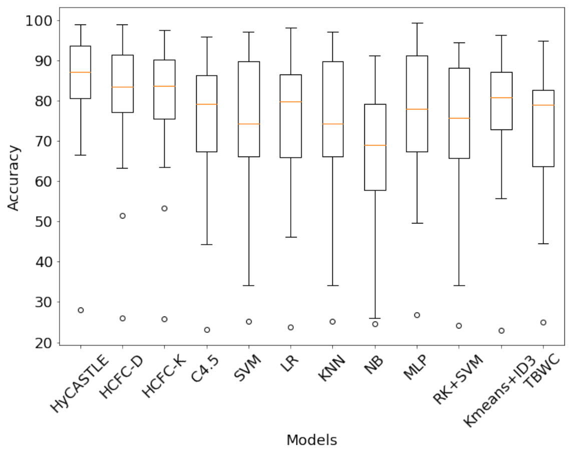

Figure 6: Box Plot comparison for the 12 models listed in Table 2.

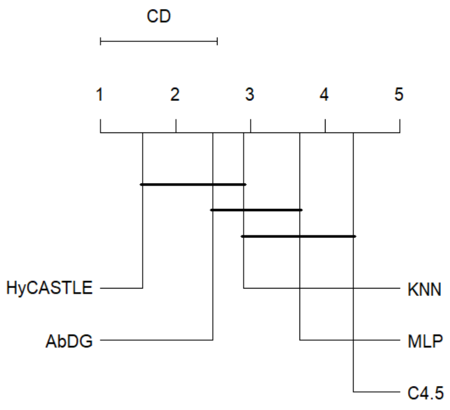

Figure 7: Critical difference plot:any two algorithms not joined by a line may be regarded as having different classification performances. The plot refers to the results shown in Table 4.

4.3. Comparison with the AbDG model

Table 4 shows the classification accuracy of HyCASTLE, AbDG, C4.5, KNN and MLP on 16 of the 35 datasets listed in Table 1.These dataset were selected in order to directly compare with Bertini et al. [12]. The three traditional supervised models (C4.5, KNN and MLP) were chosen given their good performance on the 23 dataset listed in Table 2.

As it is seen from Table 4, HyCASTLE shows a higher average classification accuracy with a lower average standard deviation than other models. Following the same procedure described in Sec. 2, we produced the Friedman, Iman-Davenport and Nemenyi *post-hoc* tests to verify whether the 5 models show statistically significant differences in classification performance. In this case $N_a = 5$ and $N_d = 16$ and thus the critical value for a confidence level of $\alpha = 0.05$ is $F(4, 60) = 2.52$.

Table 5 shows the tests results,and given that both values surpass their critical values, we can safely reject the null hypothesis.

We therefore preceded to perform the Nemenyi test (in this case the $CD = 1.56$) and, as it is seen from both Fig. 7 and 8, KNN, MLP and C4.5 show compatible performance. AbDG performance is not significantly different than that of MLP and KNN while HyCASTLE performance is significantly better then that of C4.5 and MLP but still similar to that of AbDG and KNN. As preceded in Sec. 4.2, we produced the box plot statistics

Dataset	**HyCASTLE**	**AbDG**	**C4.5**	**KNN**	**MLP**
Balance	**94.71** ± **3.20**	91.70 ± 1.30	78.60 ± 2.64	89.91 ± 3.22	90.72 ± 3.59
Blood Transfusion	**80.00** ± **2.06**	77.80 ± 3.30	76.94 ± 4.03	79.60 ± 1.61	76.27 ± 1.37
Credit Approval	88.09 ± 2.49	**89.00** ± **3.90**	84.41 ± 2.98	87.17 ± 2.23	81.98 ± 2.24
Flags	**69.47** ± **2.05**	65.50 ± 10.7	62.00 ± 3.00	67.89 ± 6.94	63.08 ± 6.61
Glass	**74.42** ± **3.28**	70.90 ± 7.70	62.44 ± 8.12	70.71 ± 3.48	68.84 ± 4.79
Haberman	77.05 ± 3.59	75.30 ± 5.80	70.30 ± 4.20	**77.37** ± **5.01**	73.87 ± 0.64
Image Segmentation	**98.68** ± **0.43**	87.30 ± 6.30	88.12 ± 5.39	96.06 ± 0.50	97.31 ± 0.40
Iris	**96.00** ± **1.33**	95.80 ± 4.50	94.15 ± 3.96	95.12 ± 1.30	89.33 ± 12.18
Pima	75.32 ± 2.09	**76.50** ± **4.60**	72.5 ± 4.92	75.19 ± 1.44	75.97 ± 3.55
Post Operative	72.41 ± 8.64	**73.30** ± **7.50**	68.21 ± 15.71	67.05 ± 7.97	65.55 ± 6.47
Soybean	81.51 ± 3.36	**94.5** ± **2.40**	91.40 ± 3.32	87.33 ± 1.51	88.52 ± 4.60
Sonar	**85.71** ± **3.98**	81.70 ± 7.60	75.12 ± 9.17	81.42 ± 6.63	79.52 ± 6.32
Waveform	**87.15** ± **1.12**	80.80 ± 5.10	72.17 ± 5.83	84.62 ± 0.53	86.52 ± 0.66
Wine	**100** ± **0.00**	98.10 ± 3.20	92.76 ± 6.11	98.33 ± 1.36	98.33 ± 1.36
WDBC	**98.42** ± **0.65**	95.00 ± 2.90	92.58 ± 3.27	98.24 ± 0.96	97.54 ± 0.86
WPBC	**80.00** ± **2.51**	79.20 ± 5.20	75.67 ± 1.98	77.94 ± 2.05	74.87 ± 4.10
Average	84.93 ± 2.55	83.28 ± 5.28	78.59 ± 5.28	83.37 ± 2.92	81.76 ± 3.73
Percentage	68.75	25.00	0.00	6.25	0.00

Table 4: Comparison of Classification Accuracy (Mean ± Standard Deviation %) between HyCASTLE, AbDG [12], C4.5 [14], KNN [16] and MLP [20]. The last two rows show respectively the average score over all datasets and the fraction of time each model was selected as the best model. In bold are highlighted the best model for each dataset.

Method	**Test Value**	**Critical Value**	**Hypothesis**
Friedman	29.74	9.48	rejected
Iman-Davenport	13.02	2.52	rejected

Table 5: Results of the Friedman and Iman-Davenport tests for testing the statistical differences between the 5 models performances shown in Table 4.

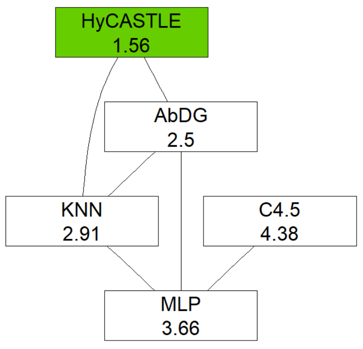

Figure 8: Detailed Critical difference plot: if two nodes are joined it means that the null hypothesis of having different performances cannot be rejected. The plot refers to the results shown in Table 4.

(see Figure 9) for the 5 models. As it can be seen HyCASTLE has a slightly higher median value that those of the other models but a higher scatter when compared to KNN.

4.4. Evaluating HyCASTLE performance with Increasing levels of Class Noise

One of the main issues presented in classification problems is the presence of high levels of noise, and thus any classification model should be constructed in order to be resilient to said noise. For example, the presence of noise in the data may produce the formation of small clusters of a particular class in an area of the parameter space where said class should not be present or could change the class boundaries between two classes making them overlap. Both this problem may cause a classification model to seriously worsen its performance. In the framework of supervised machine learning, noise can be divided into two main families [40, 41]: class noise and attribute noise. In [33],

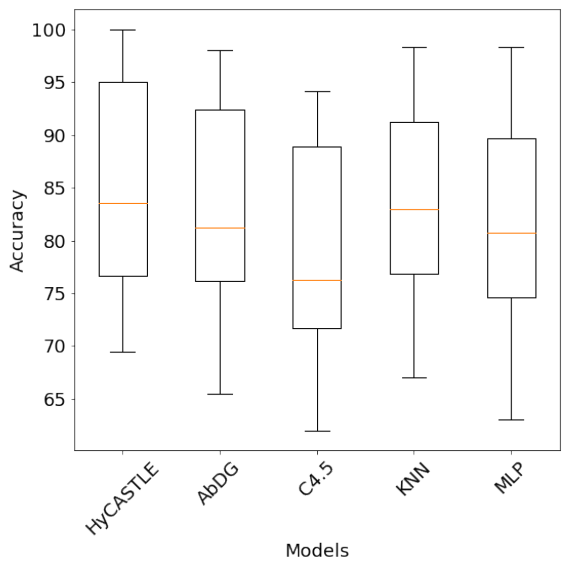

Figure 9: Box Plot comparison for the 6 models listed in Table 4.

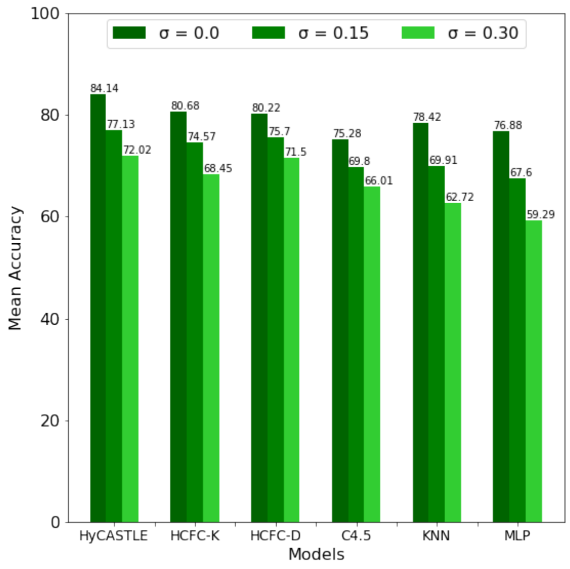

Figure 10: Average accuracy of HyCASTLE, HCFC-K, HCFC-D, C4.5, KNN and MLP on the 23 datasets listed in Table 2 when subjected to increasing levels of class noise; $\sigma = 0.$ i.e clean datasets, $\sigma = 0.15$ and $\sigma = 0.30$.

the authors demonstrated that both types of noise could have serious negative impacts on models classification performances. Nevertheless, it has been shown by the same authors that attribute noise may be more harmful then class noise, in this work, we only focus on the latter. By definition, *class noise* occurs when examples are incorrectly labelled but because this eventuality is purposely avoided in all UCI [25] datasets, we had to manually inject such noise in the data following the technique adopted by Zhu and Wu [33]. The technique uses a pairwise scheme to change the labels of the two most prominent classes in the data. Given the two classes (X, Y) and a noise level σ, an instance of class X has a $x\%$ probability of being corrupted and mislabeled as Y and so does an instance of class Y. Therefore we took the 23 datasets listed in Table 2 and repeated the experiments applying two increasing levels of class noise: $\sigma = 0.15$ and $\sigma = 0.30$. Again we chose to compare HyCASTLE classification performance with the two models proposed by Xiao et al. (HCFC-K and HCFC-D) [7] and the three standard classifiers that showed the best average accuracy on the same datasets, KNN, MLP and C4.5. HCFC-K and HCFC-D were chosen among the other hybrid models, not only because they show good and consistent performances in our previous experiment but also because, to our knowledge, their authors are the only ones in literature subjecting their proposed hybrid models to this same noise analysis. Fig. 10 shows the six average accuracy on the 23 datasets with the two levels of class noise. As it is seen, HyCASTLE shows a better average performance across both noise levels.

5. Conclusions

In this article, we presented HyCASTLE a novel hybrid classification model which utilizes a compromise between the data topology and the class labels to maximize its classification performances. Unlike other hybrid classifiers, the model employs a clustering refining loop based on both data topology and class labels to find a clustering configuration that reflects both the known knowledge of the problem (labels) and the unknown data topology. We evaluated HyCASTLE performances on 35 benchmark datasets demonstrating that HyCASTLE shows a better average classification performance than several hybrid and traditional classification models. Furthermore, we artificially injected increasing levels of class noise into 23 datasets and showed the HyCASTLE, when compared to the best models found on the previous experiments, is more resilient to class noise and shows better average classification performances. HyCASTLE has been developed in Python 3.8, and it will be made publicly available through the Scikit-Learn [34] API.

In future research, we will increase the amount of information about the clustering configuration used for prediction in fact, at this moment, a prediction is computed only through distance and cluster information is implicitly used, but it is our belief that adding information about cluster structures, purity and mutual relations explicitly through priors on the distance-based probability assignment may improve HyCASTLE performances especially in noisy environments. In this article, we didn't study HyCASTLE performance on unbalanced datasets, and we plan to investigate it. To conclude, we think that the natural evolution of HyCASTLE is to become a semi-supervised model [42, 43, 44] not only because it already performs clustering in a completely data-driven way but because it already has a way to measure the likelihood of correctness of the prediction of an unseen instance through the difference between the membership probability of the winning cluster and the remaining probability. This two features could be used to implement a self-supervised self-labelled method that should be able to solve both main issues of such models: lack of sufficient initial labelled data and the inability to deal with non-spherical data sets [45, 44, 46, 42, 47].

6. Acknowledgments

This work is entirely dedicated to the memory of our dear colleague and mentor Prof. Antonio Picariello, who left us suddenly and prematurely during the drafting of the paper and whose human and professional contribution was fundamental and decisive. We all shall badly miss his insight and his friendship.
MB acknowledges financial contributions from the agreement *ASI/INAF , Euclid ESA mission - Phase D* and the *INAF PRIN-SKA 2017 program 1.05.01.88.04*. GL acknowledges contribution from the Horizon 2020 EU program through the Marie Skłodowska-Curie grant agreement No. 721463 to the SUNDIAL ITN network. MDV acknowledge the contribution of the EUSTEMA Group who sponsored his PhD fellowship. SC acknowledges financial support from FFABR 2017 (Fondo di Finanziamento per le Attività Base di Ricerca).

References

[1] C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag, Berlin, Heidelberg, 2006.

[2] O. Laurino, R. D'Abrusco, G. Longo, G. Riccio, Astroinformatics of galaxies and quasars: a new general method for photometric redshifts estimation, MNRAS 418 (2011) 2165–2195. doi:10.1111/j.1365-2966.2011.19416.x. arXiv:1107.3160.

[3] X. Ao, P. Luo, X. Ma, F. Zhuang, Q. He, Z. Shi, Z. Shen, Combining supervised and unsupervised models via unconstrained probabilistic embedding, Information Sciences 257 (2014) 101–114. URL: https://doi.org/10.1016/j.ins.2013.08.048. doi:10.1016/j.ins.2013.08.048.

[4] J. Gao, F. Liang, W. Fan, Y. Sun, J. Han, A graph-based consensus maximization approach for combining multiple supervised and unsupervised models, IEEE Transactions on Knowledge and Data Engineering 25 (2013) 15–28. URL: https://doi.org/10.1109/tkde.2011.206. doi:10.1109/tkde.2011.206.

[5] I. Bose, X. Chen, Hybrid models using unsupervised clustering for prediction of customer churn, Journal of Organizational Computing and Electronic Commerce 19 (2009) 133–151. URL: https://doi.org/10.1080/10919390902821291. doi:10.1080/10919390902821291.

[6] C. Kaewchinporn, N. Vongsuchoto, A. Srisawat, A combination of decision tree learning and clustering for data classification, in: 2011 Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE), IEEE, 2011, pp. 363–367. URL: https://doi.org/10.1109/jcsse.2011.5930148. doi:10.1109/jcsse.2011.5930148.

[7] J. Xiao, Y. Tian, L. Xie, X. Jiang, J. Huang, A hybrid classification framework based on clustering, IEEE Transactions on Industrial Informatics 16 (2020) 2177–2188. URL: https://doi.org/10.1109/tii.2019.2933675. doi:10.1109/tii.2019.2933675.

[8] R. Rajamohamed, J. Manokaran, Improved credit card churn prediction based on rough clustering and supervised learning techniques, Cluster Computing 21 (2017) 65–77. URL: https://doi.org/10.1007/s10586-017-0933-1. doi:10.1007/s10586-017-0933-1.

[9] S. R. Gaddam, V. V. Phoha, K. S. Balagani, K-means+id3: A novel method for supervised anomaly detection by cascading k-means clustering and ID3 decision tree learning methods, IEEE Transactions on Knowledge and Data Engineering 19 (2007) 345–354. URL: https://doi.org/10.1109/tkde.2007.44. doi:10.1109/tkde.2007.44.

[10] T. Chakraborty, Ec3: Combining clustering and classification for ensemble learning, in: 2017 IEEE International Conference on Data Mining (ICDM), 2017, pp. 781–786.

[11] T. Ma, F. Wang, J. Cheng, Y. Yu, X. Chen, A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks, Sensors 16 (2016) 1701. doi:10.3390/s16101701.

[12] J. R. Bertini, M. do Carmo Nicoletti, L. Zhao, Attribute-based decision graphs: A framework for multiclass data classification, Neural Networks 85 (2017) 69–84. URL: https://doi.org/10.1016/j.neunet.2016.09.008. doi:10.1016/j.neunet.2016.09.008.

[13] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, Classification and Regression Trees, Wadsworth and Brooks, Monterey, CA, 1984.

[14] S. L. Salzberg, C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993, Machine Learning 16 (1994) 235–240. URL: https://doi.org/10.1007/bf00993309. doi:10.1007/bf00993309.

[15] U. M. Fayyad, K. B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, IJCAI (1993) 6–12.

[16] D. W. Aha, D. Kibler, M. K. Albert, Instance-based learning algorithms, Machine Learning 6 (1991) 37–66. URL: https://doi.org/10.1007/bf00153759. doi:10.1007/bf00153759.

[17] Y. Zhang, Support vector machine classification algorithm and its application, in: C. Liu, L. Wang, A. Yang (Eds.), Communications in Computer and Information Science, Springer Berlin

Heidelberg, Berlin, Heidelberg, 2012, pp. 179–186.

[18] M. Majka, naivebayes: High Performance Implementation of the Naive Bayes Algorithm in R, 2019. URL: `https://CRAN.R-project.org/package=naivebayes`, r package version 0.9.7.

[19] P. Peduzzi, J. Concato, E. Kemper, T. R. Holford, A. R. Feinstein, A simulation study of the number of events per variable in logistic regression analysis, Journal of Clinical Epidemiology 49 (1996) 1373–1379. URL: `https://doi.org/10.1016/s0895-4356(96)00236-3`. doi:10.1016/s0895-4356(96)00236-3.

[20] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer New York, 2009. URL: `https://doi.org/10.1007/978-0-387-84858-7`. doi:10.1007/978-0-387-84858-7.

[21] A. Acharya, E. Hruschka, J. Ghosh, S. Acharyya, C3e: A framework for combining ensembles of classifiers and clusterers, in: Multiple Classifier Systems, Springer Berlin Heidelberg, 2011, pp. 269–278. doi:10.1007/978-3-642-21557-5_29.

[22] M. Capó, A. Pérez, J. A. Lozano, An efficient k-means clustering algorithm for tall data, Data Mining and Knowledge Discovery 34 (2020) 776–811. URL: `https://doi.org/10.1007/s10618-020-00678-9`. doi:10.1007/s10618-020-00678-9.

[23] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proc. of 2nd International Conference on Knowledge Discovery and, 1996, pp. 226–231.

[24] P. P. Angelov, X. Gu, J. C. Príncipe, A generalized methodology for data analysis, IEEE Transactions on Cybernetics 48 (2018) 2981–2993.

[25] D. Dua, C. Graff, UCI machine learning repository, 2017. URL: `http://archive.ics.uci.edu/ml`.

[26] L. C. Freeman, A Set of Measures of Centrality Based on Betweenness, Sociometry 40 (1977) 35–41. URL: `http://dx.doi.org/10.2307/3033543`. doi:10.2307/3033543.

[27] G. Alsmeyer, Chebyshev's inequality, in: International Encyclopedia of Statistical Science, Springer Berlin Heidelberg, 2011, pp. 239–240. doi:10.1007/978-3-642-04898-2_167.

[28] C. Gini, Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche. [Fasc. I.], Studi economico-giuridici pubblicati per cura della facoltà di Giurisprudenza della R. Università di Cagliari, Tipogr. di P. Cuppini, 1912. URL: `https://books.google.it/books?id=fqjaBPMxB9kC`.

[29] M. Ankerst, M. M. Breunig, H. peter Kriegel, J. Sander, Optics: Ordering points to identify the clustering structure, in: ACM SIGMOD Record, ACM Press, 1999, pp. 49–60.

[30] A. Hinneburg, H.-H. Gabriel, Denclue 2.0: Fast clustering based on kernel density estimation, in: M. R. Berthold, J. Shawe-Taylor, N. Lavrač (Eds.), Advances in Intelligent Data Analysis VII, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 70–80.

[31] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1967) 21–27.

[32] J. B. MacQueen, Some methods for classification and analysis of multivariate observations, in: L. M. L. Cam, J. Neyman (Eds.), Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, volume 1, University of California Press, 1967, pp. 281–297.

[33] X. Zhu, X. Wu, Class noise vs. attribute noise: A quantitative study, Artificial Intelligence Review 22 (2004) 177–210. URL: `https://doi.org/10.1007/s10462-004-0751-8`. doi:10.1007/s10462-004-0751-8.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[35] S. I. Serengil, chefboost, 2019. URL: `https://github.com/serengil/chefboost`.

[36] U. M. Fayyad, K. B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning., in: IJCAI, 1993, pp. 1022–1029.

[37] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, The Annals of Mathematical Statistics 11 (1940) 86–92. URL: `http://www.jstor.org/stable/2235971`.

[38] R. Iman, J. Davenport, Approximations of the critical region of the friedman statistic, Communications in Statistics-Theory and Methods 9 (1980) 571–595.

[39] P. Nemenyi, Distribution-free Multiple Comparisons, Princeton University, 1963. URL: `https://books.google.it/books?id=nhDMtgAACAAJ`.

[40] X. Wu, Knowledge Acquisition from Databases, Tutorial Monographs in Artificial Intelligence, Intellect, Limited, 1995. URL: `https://books.google.it/books?id=UaRkE96JUssC`.

[41] J. A. Sáez, M. Galar, J. Luengo, F. Herrera, Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition, Knowledge and Information Systems 38 (2012) 179–206. URL: `https://doi.org/10.1007/s10115-012-0570-1`. doi:10.1007/s10115-012-0570-1.

[42] D. Wu, M. Shang, X. Luo, J. Xu, H. Yan, W. Deng, G. Wang, Self-training semi-supervised classification based on density peaks of data, Neurocomputing 275 (2018) 180–191. URL: `https://doi.org/10.1016/j.neucom.2017.05.072`. doi:10.1016/j.neucom.2017.05.072.

[43] H. Gan, N. Sang, R. Huang, X. Tong, Z. Dan, Using clustering analysis to improve semi-supervised classification, Neurocomputing 101 (2013) 290–298. URL: `https://doi.org/10.1016/j.neucom.2012.08.020`. doi:10.1016/j.neucom.2012.08.020.

[44] J. Li, Q. Zhu, Q. Wu, D. Cheng, An effective framework based on local cores for self-labeled semi-supervised classification, Knowledge-Based Systems 197 (2020) 105804. URL: `https://doi.org/10.1016/j.knosys.2020.105804`. doi:10.1016/j.knosys.2020.105804.

[45] J. Li, Q. Zhu, Q. Wu, A self-training method based on density peaks and an extended parameter-free local noise filter for k nearest neighbor, Knowledge-Based Systems 184 (2019) 104895. URL: `https://doi.org/10.1016/j.knosys.2019.104895`. doi:10.1016/j.knosys.2019.104895.

[46] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (2014) 1492–1496. URL: `https://doi.org/10.1126/science.1242072`. doi:10.1126/science.1242072.

[47] Y. Wang, X. Xu, H. Zhao, Z. Hua, Semi-supervised learning based on nearest neighbor rule and cut edges, Knowledge-Based Systems 23 (2010) 547–554. URL: `https://doi.org/10.1016/j.knosys.2010.03.012`. doi:10.1016/j.knosys.2010.03.012.