information

MDPI

Article

Prototype Selection for Multilabel Instance-Based Learning [†]

Panagiotis Filippakis [1,*], Stefanos Ougiaroglou [1] and Georgios Evangelidis [2]

[1] Department of Information and Electronic Engineering, School of Engineering, International Hellenic University, 57400 Thessaloniki, Greece; stoug@ihu.gr
[2] Department of Applied Informatics, School of Information Sciences, University of Macedonia, 156 Egnatia Street, 54636 Thessaloniki, Greece; gevan@uom.gr
[*] Correspondence: filipana1@iee.ihu.gr

[†] This paper is an extended version of our paper published in 27th International Database Engineered Application Symposium, IDEAS 2023, Heraklion, Greece, 5–7 May 2023.

Abstract: Reducing the size of the training set, which involves replacing it with a condensed set, is a widely adopted practice to enhance the efficiency of instance-based classifiers while trying to maintain high classification accuracy. This objective can be achieved through the use of data reduction techniques, also known as prototype selection or generation algorithms. Although there are numerous algorithms available in the literature that effectively address single-label classification problems, most of them are not applicable to multilabel data, where an instance can belong to multiple classes. Well-known transformation methods cannot be combined with a data reduction technique due to different reasons. The Condensed Nearest Neighbor rule is a popular parameter-free single-label prototype selection algorithm. The IB2 algorithm is the one-pass variation of the Condensed Nearest Neighbor rule. This paper proposes variations of these algorithms for multilabel data. Through an experimental study conducted on nine distinct datasets as well as statistical tests, we demonstrate that the eight proposed approaches (four for each algorithm) offer significant reduction rates without compromising the classification accuracy.

Keywords: data reduction techniques; instance reduction; multilabel classification; prototype selection; instance-based classification; binary relevance; CNN; IB2; BR*k*NN

Citation: Filippakis, P.; Ougiaroglou, S.; Evangelidis, G. Prototype Selection for Multilabel Instance-Based Learning. *Information* **2023**, *14*, 572. https://doi.org/10.3390/info14100572

Academic Editor: Peter Revesz

Received: 31 July 2023
Revised: 13 October 2023
Accepted: 17 October 2023
Published: 19 October 2023

1. Introduction

Multilabel classification [1] involves predicting multiple potential classes or labels for a single instance, while single-label classification focuses on assigning only one class to each instance. Multilabel classification is commonly employed to classify diverse forms of data, such as images, books, artists, music, videos and movies. For instance, a movie can be classified as both "crime" and "adventure", a text can cover many different topics and a music track may encompass multiple genres or moods. Multilabel classification can be characterized as a generalization of single-label classification, where a classifier is capable of handling scenarios where multiple labels may be applicable to a single instance. This extension allows for more versatile predictions as it accommodates cases where instances can belong to multiple classes at the same time.

The *k*-Nearest Neighbors (*k*-NN) [2] classification algorithm serves as a typical illustration of a lazy or instance-based classifier. It operates by retrieving the *k*-nearest neighbors of an unclassified instance and employing a majority voting approach to assign a classification. In simpler terms, the unclassified instance is assigned to the most prevalent class among the classes of the retrieved *k*-nearest neighbors. This classifier is renowned for its simplicity, ease of implementation and robust classification performance, making it valuable for both single-label and multilabel classification tasks. Nonetheless, it comes with a drawback of high computational cost due to the need to calculate the distances between each unclassified instance and all instances in the training set.

Hence, the size of the training set plays a vital role in instance-based classification. While a large training set yields higher classification accuracy, it also entails increased computational costs. To expedite the *k*-NN classifier, it becomes necessary to mitigate its memory and CPU requirements by reducing the training set's size. In single-label classification tasks, one approach is to employ a data reduction technique (DRT) capable of reducing either the number of training instances or attributes [3]. This paper specifically focuses on DRTs from the perspective of instance reduction. The objective of this paper is to achieve efficient *k*-NN classification on multilabel data by decreasing the training set's size without compromising accuracy.

DRTs encompass two categories, namely prototype selection (PS) [4] and prototype generation (PG) [5]. In practical terms, these techniques serve as data pre-processing tasks aimed at replacing the original training dataset with a smaller subset known as the "condensing set". Utilizing the condensing set enables the *k*-NN classifier to achieve comparable accuracy to using the full training dataset but with significantly reduced computational costs. PS algorithms choose specific instances, or prototypes, from the original training set, whereas PG algorithms generate prototypes by summarizing similar training instances belonging to the same class. The fundamental concept underlying many DRTs is that only training instances in close proximity to class decision boundaries, in terms of a Euclidean metric space, are crucial for classification tasks. Those training instances situated within the "internal" area of a class, far away from decision boundaries, can be safely removed without compromising classification accuracy. Consequently, DRTs aim to select or generate an adequate number of prototypes that reside near the decision boundary areas for each class. The majority of DRTs primarily focus on single-label classification problems.

It is worth mentioning that a subcategory of PS algorithms focuses on noise removal and operates differently from other PS and PG algorithms. These algorithms are designed to eliminate noise and smooth the decision boundaries between discrete classes. As a result, they create an edited training set that leads to accuracy gains in the classification process.

The label powerset (LP) transformation technique [1] offers a straightforward solution for employing a DRT in multilabel classification tasks. LP transforms a multilabel dataset into a single-label dataset by considering each label combination, or labelset, as a separate class. However, it is important to acknowledge that LP is suitable only when the number of labels and potential labelsets is limited and there are ample instances available for each labelset. In situations where the number of label combinations becomes excessively large, the reduction rate may not be sufficient, resulting in inadequate representation of certain combinations. Moreover, the total count of distinct label combinations can grow exponentially, giving rise to scalability issues.

Binary relevance (BR) is a popular transformation technique that addresses multilabel classification problems by transforming them into single-label classification problems. Essentially, BR involves converting the original multilabel problem into multiple independent binary classification problems. In order to predict the labels associated with an unclassified instance, a separate classifier is required for each available label. When BR is combined with the *k*-NN classifier, it is referred to as BR*k*NN [6]. This combination proves effective because the *k*-NN classifier is a lazy classifier that does not construct a classification model. During the classification of an instance x using BR*k*NN, the algorithm searches for the *k*-nearest neighbors to x, just like in the case of single-label *k*-NN. Subsequently, the voting procedure of the nearest neighbors is repeated individually for each label.

When a data reduction technique (DRT) is applied before utilizing the *k*-NN classifier, the classifier transitions from being lazy to eager. In such scenarios, the condensing set becomes the classification model. However, in the context of multilabel classification, the application of BR (binary relevance) to construct a condensing set for each label undermines the objective of data reduction. On one hand, the goal of data reduction is not achieved. On the other hand, since the *k*-NN classifier must search for nearest neighbors within each specific condensing set to make predictions for individual labels, the computational cost remains high. Hence, combining BR with a DRT becomes infeasible due to the presence of multiple binary condensing sets. This highlights the necessity of modifying DRTs to effectively handle multilabel datasets, which serves as the motivation for the current research work.

The Condensed Nearest Neighbor (CNN) rule [7] stands as the oldest prototype selection (PS) algorithm for single-label classification tasks. It operates by eliminating training instances that reside far from the decision boundaries, and this is achieved by running over the training data multiple times. Instance-Based Learning 2 (IB2) [8] is the one-pass variation of CNN and has the same motivation with that of CNN. IB2 involves an extremely low pre-processing computational cost to build the condensing set. Both CNN and IB2 are popular parameter-free PS algorithms. However, they are inappropriate for multilabel data.

The objective of the present paper is to extend the applicability of CNN and IB2 by introducing variations suitable for multilabel classification problems. In [9], we proposed three multilabel variations of CNN. Here, we extend the previous work by adding one more variation of CNN, which is based on Levenshtein distance [10], and by introducing the multilabel version of IB2 with the four variations. Thus, the contributions of the paper are summarized as follows:

- We propose four variations of CNN and IB2 that are suitable for multilabel classification problems.
- One of the variations uses a novel adaptation of Levenshtein distance for multilabel instances.
- We conduct an experimental study using nine multilabel datasets and complement it with corresponding statistical tests of significance. The study reveals that the proposed variations offer significant reduction rates without compromising the classification accuracy.

The rest of this paper is organized as follows. Section 2 presents the related work in data reduction on multilabel datasets. Section 3 presents the CNN algorithm for single-label classification problems. Section 4 presents the Instance-Based Two-Step (IB2) algorithm for single-label classification problems. Section 5 describes the proposed variations of CNN and IB2 for multilabel classification problems that use the BR*k*NN multilabel classifier. Section 6 presents the experimental study that compares the proposed algorithms. Finally, Section 7 concludes the paper.

2. Related Work

The majority of publications focused on multilabel problems tend to discuss classification algorithms rather than approaches aimed at reducing computational costs associated with large multilabel training sets. There are also many attempts to offer programming APIs [11] and environments [12] specific for multilabel classification. Limited research is available on data reduction techniques specifically tailored for such datasets. In this section, we examine the scarce relevant literature.

In [13], the authors propose a PS algorithm designed for multilabel datasets. This algorithm aims to eliminate noise during the editing process and achieve balanced class decision boundaries. Inspired by the Edited Nearest Neighbor rule (ENN-rule) [14], the authors suggest an under-sampling method for addressing imbalanced training sets.

Another article, [15], introduces a prototype selection editing algorithm based on ENN-rule. The algorithm utilizes the Hamming loss metric to identify noisy training instances. The concept is straightforward: instances with high Hamming loss are considered for elimination due to their proximity to decision boundaries, similar to ENN-rule.

In [16], the authors make the first attempt to adapt PS algorithms to multilabel problems. Their proposed algorithms are based on local sets [17] and the LP transformation technique [1]. In single-label problems, the local set of an instance refers to the largest set of instances centered around it, all belonging to the same class. The authors argue that, in multilabel datasets, a local set does not necessarily need to contain instances with the exact same labelsets; they may have slightly different labelsets. The authors calculate the Hamming loss over the labelsets to measure the differences, and the distance between two instances is determined using the Hamming loss of their labelsets. If the distance exceeds a specified threshold, the instances are classified as belonging to different "classes".

The multilabel prototype selection with Co-Occurrence and Generalized Condensed Nearest Neighbor (CO-GCNN) was proposed in [18]. CO-GCNN captures label correlation by computing the co-occurrence frequency of label pairs and subsequently segregating the initial data into positive and negative categories. The single-label generalized CNN [19] is performed in order to produce the condensing set. In effect, CO-GCNN transforms the multilabel classification problem into a single-label classification problem. It leverages the benefit of incorporating pairwise label correlations as a constraint during the data transformation step. The authors state that the incorporation of pairwise label correlations enables the chosen prototypes to more accurately represent the original dataset.

In Ref. [20], a simple multilabel prototype selection algorithm based on clustering is proposed. The proposed methodology uses a clustering algorithm as a PS algorithm and then the well-known Multilabel *k*-Nearest Neighbor algorithm (ML-KNN) [21] performs the labels prediction.

The article [22] explores the use of single-label prototype selection algorithms along with binary relevance (BR), label powerset (LP) and other transformation techniques. For BR and its variants, the proposed strategy generates individual single-label training sets for different label types. Each training set undergoes a PS algorithm to create a condensing set specific to each label. Instances receive votes each time they are selected, and the accumulated votes form a single vector. Instances with votes surpassing a predefined threshold are selected, resulting in a complete condensing set.

The work presented in [23] does not introduce a DRT. However, the proposed method uses a PS algorithm as an intermediate step. The authors argue that PS leads to accuracy loss. Their method aims to combine the classification accuracy of retaining the original training set with the time efficiency of a PS method. More specifically, the authors propose a three-phase strategy for multilabel classification: initially, a PS algorithm is employed on the complete training set, generating a single-label condensed set. This operation is performed once as a pre-processing step. When an instance x is presented and must be classified, the proposed method selects a reduced set of labels as potential hypotheses, considering only the condensed set. In effect, the condensing set works as a recommender system that recommends the labels where x belongs to. The authors suggest picking the c nearest classes to input x, where c is a user-specified parameter. The final classification is performed by k-NN, employing a dynamically formed subset of the original training set limited to the c labels identified in the previous step.

In [24], a data pre-processing technique to improve label distribution learning (LDL) [25] algorithms is proposed. LDL is a general learning framework that assigns an instance to a distribution over a set of labels. Specifically, the proposed method is called ProLSFEO-LDL and combines prototype selection and label-specific feature learning. The paper proposes an evolutionary algorithm adapted to the specificities of LDL, aiming to optimize the initial solution to meet desired expectations.

In [26], the authors introduce an attempt to reduce multilabel datasets using homogeneous clustering. The algorithm, known as Multilabel Reduction through Homogeneous Clusters (MLRHC), is an adaptation of the single-label prototype generation algorithm RHC. MLRHC applies K-means clustering iteratively to produce homogeneous clusters, which are then replaced by their center. In MLRHC, a cluster is considered homogeneous if all instances within the cluster share at least one common label. The initial dataset is clustered using the existing labels as initial means, with the number of clusters matching the number of existing labels. Homogeneous clusters are replaced by their center, which is assigned the common label along with any label appearing in at least half of the cluster's instances. Similarly, in [27], the authors extend their previous work by proposing a variant called Multilabel Reduction by Space Partitioning (MLRSP3), which also relies on the concept of homogeneity based on instances sharing at least one common label. MLRSP3 is based on the RSP3 PG algorithm [28] and starts with a non-homogeneous cluster, selects the two farthest instances and divides the training set into two clusters by assigning instances to the closest farthest instance. This process continues until all clusters become homogeneous. Like MLRHC, the center of each homogeneous cluster in MLRSP3 becomes a multilabel prototype labeled with the common label and any label appearing in at least half of the instances within the cluster.

Quite similar work is presented in [29]. More specifically, the authors propose adaptations of the Chen and Jozwik multiclass PG algorithm [30] and of its descendants, RSP1, RSP2 and RSP3 [28], to the multilabel case. The proposed adaptations are evaluated on three multilabel NN-based classifiers using 12 datasets of varying domains and sizes, along with artificially induced noise scenarios. The results show high efficiency and classification performance, especially in noisy scenarios.

The paper presented in [31] does not introduce a PS or a PG algorithm. However, it deals with the high computational cost in multilabel classification. Specifically, the paper introduces a novel approach for multilabel classification by leveraging hypergraph spectral learning. Hypergraphs, which generalize traditional graphs by allowing edges to be arbitrary non-empty subsets of vertices, are employed to capture high-order relations among labels. The proposed formulation leads to an eigenvalue problem, which may be computationally expensive for large-scale datasets. To address this, the paper presents an approximate formulation that reduces computational complexity while maintaining competitive classification performance.

Figure 1 summarizes the presented works in a form of a hierarchy. More specifically, the presented works can be categorized into three main categories. The first category includes the paper related with the PS algorithms and the adaption of them. The works related to adaptation of PG algorithms belong to the second category. The methods that belong to the third category cannot be characterized as either PS or PG. However, they are able to speed up the multilabel classification tasks, which is the goal of the PS and PG algorithms.

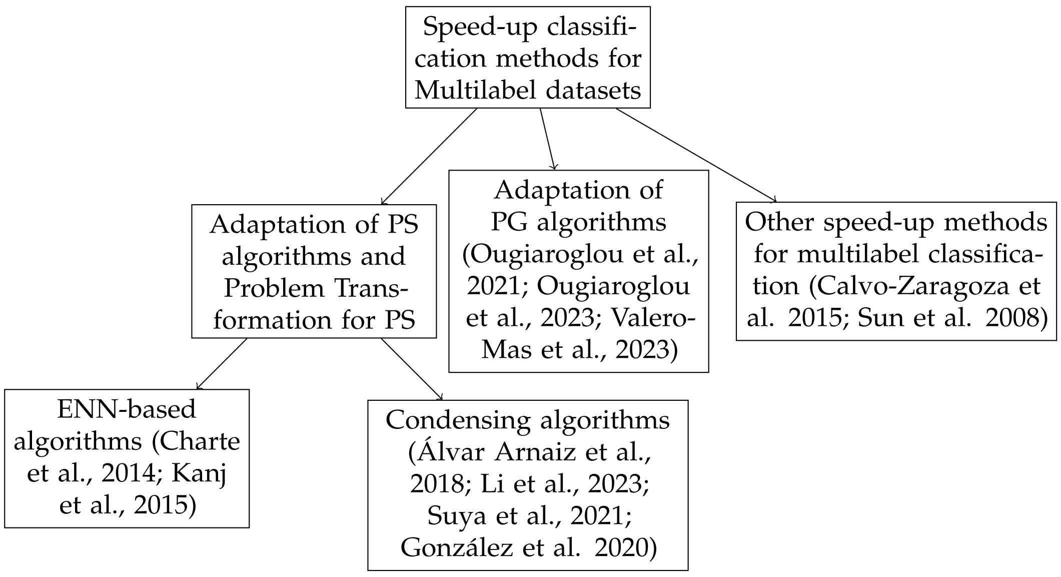
Speed-up classification methods for Multilabel datasets
Adaptation of PS algorithms and Problem Transformation for PS
Adaptation of PG algorithms (Ougiaroglou et al., 2021; Ougiaroglou et al., 2023; Valero-Mas et al., 2023)
Other speed-up methods for multilabel classification (Calvo-Zaragoza et al. 2015; Sun et al. 2008)
ENN-based algorithms (Charte et al., 2014; Kanj et al., 2015)
Condensing algorithms (Álvar Arnaiz et al., 2018; Li et al., 2023; Suya et al., 2021; González et al. 2020)

Figure 1. Hierarchy of the presented algorithms and methods: Charte et al., 2014 [13]; Kanj et al., 2015 [15]; Álvar Arnaiz et al., 2018 [16,22]; Li et al., 2023 [18]; Suya et al., 2021 [20]; González et al., 2020 [24]; Ougiaroglou et al., 2021 [26]; Ougiaroglou et al., 2023 [27]; Valero-Mas et al., 2023 [29]; Calvo-Zaragoza et al., 2015 [23]; Sun et al., 2008 [31].

3. The Single-Label Condensed Nearest Neighbor Rule

In Section 1, the Condensed Nearest Neighbor (CNN) rule [7] is discussed as the first and most widely utilized PS algorithm. CNN is a parameter-free single-label PS algorithm that constructs its condensing set by iteratively examining the training data.

CNN involves two storage areas, namely the Condensing Set (CS) and the Training Set (TS). Initially, the TS contains the entire training set while the CS is empty. The process begins by randomly selecting an instance from the TS and transferring it to the CS. Each instance $x \in$ TS is then compared to the instances currently stored in the CS.

Specifically, for each instance $x \in TS$, CNN identifies its nearest neighbor (1-NN) within the current CS using the Euclidean distance. If x is correctly classified by its nearest neighbor in the CS, it remains in the TS. However, if x is misclassified, it is removed from the TS and added to the CS. This process continues until all instances $x \in TS$ have been considered. Subsequently, the algorithm proceeds to the next scan of the TS.

The algorithm terminates when, during a complete scan of the TS, no instances are transferred from the TS to the CS, indicating that all instances in the TS are correctly classified based on the content of the CS. Algorithm 1 provides the pseudo-code representation of the CNN algorithm.

It is worth mentioning that, during the first algorithm iteration, CS is not empty. It contains a randomly selected instance (see line 2 in Algorithm 1). Therefore, for all examined instances of TS, there is always a nearest neighbor in CS.

Algorithm 1 CNN

```
Input: $TS$
Output: $CS$
1:  $CS \leftarrow \varnothing$
2:  randomly pick an instance of TS and move it to CS
3:  repeat
4:    $stop \leftarrow TRUE$
5:    for each $x \in TS$ do
6:      $NN \leftarrow$ nearest neighbor of $x$ in $CS$ using Euclidean distance
7:      if $NN_{class} \neq x_{class}$ then
8:        $CS \leftarrow CS \cup \{x\}$
9:        $TS \leftarrow TS - \{x\}$
10:       $stop \leftarrow FALSE$
11:     end if
12:   end for
13: until $stop == TRUE$ {no move during a pass of TS}
14: discard TS
15: return $CS$
```

The fundamental principle of the CNN algorithm is to include incorrectly classified instances in the Condensing Set (CS) since they are considered border instances located near decision boundaries. CNN ensures that each removed instance from the Training Set (TS) can be correctly classified using the information contained in the CS set. CNN's lack of parameters is a significant advantage. The condensing set is built without the need of any user-specified input parameter. Thus, costly computational parameter tuning procedures are avoided. On the other hand, there are some disadvantages to consider:

- Variability in Results: Running the CNN algorithm multiple times on the same TS may produce different condensing sets due to variations in the randomly selected initial instance (line 2 of Algorithm 1) or differences in the order in which TS instances are examined (line 5 of Algorithm 1).
- Memory Requirements: CNN is a memory-based algorithm, meaning that all instances need to reside in main memory during its execution.

- Computational Cost: The CNN algorithm requires multiple passes over the training set.

In terms of quality, the CNN algorithm operates as follows: if the underlying densities of different classes have minimal overlap, indicating a low Bayes risk, CNN tends to select instances located close to the possibly fuzzy boundary between classes. Instances deeply embedded within a class are unlikely to be transferred to the CS since they are correctly classified. However, if the Bayes risk is high, the CS will essentially contain almost every instance from the initial TS set, resulting in negligible sample size reduction.

4. The IB2 Algorithm

Aha et al. [8] introduced a set of instance-based learning algorithms. Among these algorithms, IB1 (Instance-Based Learning) served as a baseline and was essentially equivalent to the 1-NN algorithm.

The IB2 algorithm operates incrementally by initially having an empty set, CS, and adding each instance from TS to CS if it is misclassified by the instances already present in CS. Algorithm 2 provides the pseudo-code representation of the IB2 algorithm.

Algorithm 2 IB2

```
Input: $TS$
Output: $CS$
1:  $CS \leftarrow \varnothing$
2:  randomly pick an instance of TS and move it to CS
3:  for each $x \in TS$ do
4:      $NN \leftarrow$ nearest neighbor of $x$ in $CS$ using Euclidean distance
5:      if $NN_{class} \neq x_{class}$ then
6:          $CS \leftarrow CS \cup \{x\}$
7:          $TS \leftarrow TS - \{x\}$
8:      end if
9:  end for
10: discard TS
11: return $CS$
```

IB2 bears similarities to CNN-rule but it does not repeat the process after the first pass through the training set. As a result, IB2 does not guarantee the correct classification of all remaining instances in the TS. In effect, IB2 is a one-pass version of CNN.

Like CNN, IB2 aims to retain border instances in CS while eliminating internal instances that are surrounded by instances belonging to the same class. Similar to the CNN algorithm, IB2 is highly sensitive to noise because erroneous instances are often misclassified, resulting in the preservation of noisy instances, while more reliable instances are removed. The benefits and characteristics of IB2 algorithm are

- Since IB2 avoids multiple passes over training data, is quite faster than CNN.
- The condensing set obtained from IB2 is generally smaller than that of CNN, leading to faster classification and reduced storage requirements.
- IB2 supports incremental learning, where new instances can be added to the condensing set without requiring complete retraining of the algorithm.
- The decision boundary revision step allows IB2 to adapt to new instances and adjust the reduced training set accordingly.

5. The Proposed Algorithms

As discussed in Section 1, traditional data reduction algorithms are not suitable for use with the binary relevance transformation method in multilabel data. Applying data reduction in conjunction with the binary relevance transformation would lead to the creation of numerous condensing sets, one for each label.

In this section, we introduce variations of the CNN and IB2 algorithms that are designed for multilabel datasets. These proposed algorithms are named

- Multilabel CNN Hamming Distance and Multilabel IB2 Hamming Distance (MLCNN-H and MLIB2-H);
- Multilabel CNN Jaccard Distance and Multilabel IB2 Jaccard Distance (MLCNN-J and MLIB2-J);
- Multilabel CNN Levenshtein Distance and Multilabel IB2 Levenshtein Distance (MLCNN-L and MLIB2-L) and
- Multilabel CNN Binary Relevance and Multilabel IB2 Binary Relevance (MLCNN-BR and MLIB2-BR).

5.1. MLCNN-H and MLIB2-H

The MLCNN-H algorithm is based on a similar principle to CNN, with the idea that an instance with a significantly different labelset compared to its nearest neighbor should be included in the multilabel condensing set. To achieve this, MLCNN-H requires a method to measure the distance or difference between multilabel instances and a mechanism to determine when two instances are considered different or similar.

For that purpose, MLCNN-H uses Hamming distance. The Hamming distance between two labelsets is the number of positions at which the corresponding labels do not match.

Definition 1 (Hamming Distance)**.** *Given two labelsets u and v, each of length n, their Hamming distance is the total number of positions where their labels do not match:*

$$HD(u,v) = Cardinality(\{i : u_i \neq v_i, i = 1, \ldots, n\})$$

In MLCNN-H, the labelset of an instance is represented as a sequence of binary values (0 or 1), where 0 indicates that the instance does not belong to the corresponding label and 1 indicates that it does. To compute the Hamming distance (HD) between two instances, the labelsets are compared using an XOR operation to count the number of differing labels. To express HD in the [0,1] interval, this count is then divided by the length of a labelset, which represents the total number of labels in the dataset. Consequently, when HD is zero, the two instances have identical labelsets, while an HD of one indicates completely different labelsets. For example:

- HD(110001, 110001) = 0/6 = 0
- HD(110001, 001110) = 6/6 = 1
- HD(110001, 000010) = 4/6 = 0.33
- HD(110001, 100010) = 3/6 = 0.5

MLCNN-H incorporates the concept of label density in the dataset. The label density is calculated as the average number of labels per instance divided by the number of distinct labels in the dataset [32]. MLCNN-H considers the labelsets of examined instances to be significantly different if their HD is greater than the dataset density.

Similar to the single-label CNN algorithm, MLCNN-H utilizes two sets: the Condensing Set (CS) and the Training Set (TS). Initially, TS contains the complete training set, while CS is empty. MLCNN-H randomly selects an instance from TS and transfers it to CS. For each instance $x \in TS$, the algorithm finds the nearest neighbor (e.g., y) within the CS. Then, MLCNN-H calculates the Hamming distance (HD) metric between x and y, quantifying the difference in their labelsets. If HD is greater than the dataset density, x is removed from TS and added to CS; otherwise, it remains in TS. Once all instances $x \in TS$ have been examined, the process continues with subsequent scans of the remaining instances in TS. MLCNN-H terminates when no transfers from TS to CS are made during a complete pass over TS. In each next scan over the remaining instances in TS, more instances move from TS to CS.

Similar to MLCNN-H, the MLIB2-H algorithm follows exactly the same process as the previous one, with the only difference being that the process stops after examining all instances in TS in one pass. In effect, MLIB2-H is the one-pass version of MLCNN-H. Therefore, MLIB2-H is quite faster and achieves higher reduction rates than MLCNN-H.

5.2. MLCNN-J and MLIB2-J

MLCNN-J and MLIB2-J employ the Jaccard distance for asymmetric binary attributes, where the presence of a label is considered more important than its absence.

Definition 2 (Jaccard Distance for asymmetric binary attributes)**.** *Given two labelsets u and v, each of length n, we define:*

$$A = Cardinality(\{i : u_i = v_i = 1, i = 1, \dots, n\})$$

$$B = Cardinality(\{i : u_i = 1 \land v_i = 0, i = 1, \dots, n\})$$

$$C = Cardinality(\{i : u_i = 0 \land v_i = 1, i = 1, \dots, n\})$$

$$D = Cardinality(\{i : u_i = v_i = 0, i = 1, \dots, n\})$$

Then, their Jaccard distance for asymmetric binary attributes is the percentage of the number of positions where their labels do not match over the total number of positions where at least one of the labelsets has an appearing label. In other words, D is not taken into consideration:

$$JD(u, v) = \frac{B + C}{A + B + C}$$

Thus, matching zeros (absent labels) are disregarded when calculating the distance between two labelsets. For example:

- JD(110001, 110001) = 0/3 = 0
- JD(110001, 001110) = 6/6 = 1
- JD(110001, 000010) = 4/4 = 1
- JD(110001, 100010) = 3/4 = 0.75

MLCNN-J and MLIB2-J are variations of MLCNN-H and MLIB2-H, respectively. Both are based on the idea of selecting instances with labelsets that significantly differ from their nearest neighbors in the CS as prototypes. However, MLCNN-J and MLIB2-J differ from MLCNN-H and MLIB2-H in two key aspects. Firstly, instead of using Hamming distance, MLCNN-J and MLIB2-J employ Jaccard distance (JD) as the dissimilarity metric. Secondly, MLCNN-J and MLIB2-J introduce a different JD threshold to determine the extent to which two instances differ.

The JD threshold plays a crucial role in determining which instances are considered different enough to be included in the CS. Initially, MLCNN-J and MLIB2-J consider instances with fewer than half the labels in common (JD threshold of 0.5) as different. This means that instances sharing at least half the labels are deemed similar and not added to the CS. However, in order to increase the reduction rates, higher JD threshold values were explored during experimentation. Ultimately, two JD threshold values were chosen for testing: 0.5 and 0.75.

By employing Jaccard distance and adjusting the JD threshold, MLCNN-J and MLIB2-J aim to identify instances that significantly differ from their nearest neighbors and include them as prototypes in the CS. The choice of the JD threshold affects the reduction rates, with higher thresholds potentially leading to greater reductions by considering more instances as similar to their nearest neighbors and excluding them from the CS.

The MLIB2-J algorithm follows the exact same procedure as the MLCNN-J but with a distinct difference: it conducts a single complete iteration on the training set and subsequently terminates. Thus, it achieves higher reduction rates than MLCNN-J.

5.3. MLCNN-L and MLIB2-L

MLCNN-L and MLIB2-L are the third pair of multilabel variations of CNN. MLCNN-L and MLIB2-L utilize the Levenshtein distance metric. The Levenshtein distance serves as a quantification of dissimilarity between two sets.

The Levenshtein distance represents the minimum number of edit operations needed to convert one string into another. These edit operations encompass insertions, deletions and substitutions. Among the family of distance metrics known as edit distance, the Levenshtein distance stands out as one of the most widely used and popular metrics.

For instance, for the transformation of the string "COVID" to the string "MOVING", three operations are required. Hence, the Levenshtein distance between these two strings is three. More specifically, C is substituted by M, D is substituted by N and, finally, G is inserted.

Properties of the Levenshtein distance include:

1. Non-Negativity: The Levenshtein distance is always non-negative.
2. Symmetry: The distance between "S" and "T" is the same as the distance between "T" and "S".
3. Identity: The distance between a string and itself is always zero.
4. Triangle Inequality: For any three strings "S", "T" and "W", the distance from "S" to "W" is no greater than the sum of the distances from "S" to "T" and from "T" to "W".
5. Substructure Optimality: The optimal solution for the overall Levenshtein distance can be obtained by combining optimal solutions to the subproblems (i.e., the prefix substrings) of "S" and "T" [33].

MLCNN-L and MLIB2-L are modified versions of MLCNN-J and MLIB2-J that focus on selecting instances with labelsets that exhibit substantial differences from their nearest neighbors in the CS as prototype examples. However, MLCNN-L and MLIB2-L distinguish themselves from MLCNN-J and MLIB2-J in two ways. Firstly, they replace the use of Jaccard distance with Levenshtein distance as the dissimilarity metric. Secondly, MLCNN-L and MLIB2-L introduce a distinct Levenshtein distance threshold to determine the degree of dissimilarity between two instances. In MLCNN-L and MLIB2-L, the concept of label cardinality is incorporated. Label cardinality (LC) of a dataset refers to the average number of labels per instance in the dataset.

In MLCNN-L and MLIB2-L, the labelsets of examined instances are considered significantly different if the Levenshtein distance between them exceeds half of the label cardinality (LC). Therefore, MLCNN-L and MLIB2-L utilize the Levenshtein distance (LV) and consider instances for which $LV > \frac{LC}{2}$ to differ significantly. The goal is to include these instances as prototypes in the CS.

Let us now illustrate how the Levenshtein distance is computed for two instances. Using the binary representation of the labelsets of the instances, we perform an on-the-fly mapping to an ASCII string. This is accomplished by mapping label positions to a fixed sorted sequence of ASCII characters. For example, assuming that there are six labels in total, Positions 1–6 are mapped to characters A through F. Hence, 011010 is mapped to BCE, whereas 111001 is mapped to ABCF. The examples below demonstrate the use of Levenshtein distance on the mapped labelsets:

- LEV(110001, 110001) = LEV(ABF, ABF) = 0
- LEV(110001, 001110) = LEV(ABF, CDE) = 3
- LEV(110001, 000010) = LEV(ABF, E) = 3
- LEV(110001, 100010) = LEV(ABF, AE) = 2

We can take advantage of the fact that, by design, the resulting strings are sorted sequences of characters and compute Levenshtein distances directly on the binary labelsets, i.e., without mapping them to strings.

Like in the case of Jaccard distance for asymmetric binary attributes, our method for mapping binary labelsets to strings disregards non-appearing labels, i.e., 0 to 0 matches.

We define a matching substring pair of two labelsets to consist of 1s only, whereas non-matching substring pairs of two labelsets are all the remaining cases. For example, let us take labelsets x = 11001001 and y = 11110001. We can express these two labelsets as a sequence of matching and non-matching substring pairs as follows: (11, 11), (00100, 11000) and (1, 1). The first and third substring pairs are matching, whereas the second substring pair is non-matching. It is obvious that matching substring pairs correspond to identical strings, while non-matching substring pairs correspond to strings without a single common character.

In our example, since mapped(x) = ABEH and mapped(y) = ABCDH, the corresponding mapped substring pairs are (AB, AB), (E, CD) and (H, H). Observe that F and G are missing from both mapped labelsets. By definition, the Levenshtein distance of identical strings is zero and of strings without any common characters is the length of the longest string. Thus, to calculate the Levenshtein distance of the mapped labelsets of our example, we sum the Levenshtein distances of their non-matching substring pairs. In our example, this is pair (E, CD) and the distance is 2. Using the original binary labelsets, the Levenshtein distance of a non-matching substring pair is the maximum number of 1s among the two substrings. In our example, the Levenshtein distance of (00100, 11000) is 2.

Like the previous presented variations, the MLIB2-L algorithm adheres to the same procedure as MLCNN-L but with a distinction: it carries out a single full iteration on the training set and then terminates.

5.4. MLCNN-BR and MLIB2-BR

MLCNN-BR and MLIB2-BR take a different approach compared to MLCNN-H and MLIB2-H, MLCNN-J and MLIB2-J, MLCNN-L and MLIB2-L. Both MLCNN-BR and MLIB2-BR begin by using the binary relevance transformation method to transform the multilabel problem with $|L|$ labels into $|L|$ single-label problems. Each label of the training set is processed separately using the CNN or IB2 algorithm. This results in the creation of multiple CSs, with each set corresponding to a specific label. For example, if the initial training set has ten labels, MLCNN-BR and MLIB2-BR generate ten CSs.

Each CS contains prototypes labeled as 1 (indicating that the instance belongs to the corresponding label) or 0 (indicating that the instance does not belong to the corresponding label). From each condensing set, MLCNN-BR and MLIB2-BR select only the prototypes with a label value of 1, discarding the prototypes with a label value of 0. Finally, MLCNN-BR and MLIB2-BR merge all the CS to create the final multilabel CS.

To illustrate how MLCNN-BR works, consider the example of running CNN or IB2 for each label on a two-dimensional training dataset with two labels. Suppose two CSs, as shown in Tables 1 and 2, are derived. The first CS contains six prototypes labeled as “1”, while the second CS contains four prototypes labeled as “1”. The final multilabel CS constructed by MLCNN-BR or MLIB2-BR, as shown in Table 3, contains eight prototypes.

In the final multilabel CS, prototypes (1, 1), (1, 8), (4, 5) and (9, 1) originate exclusively from the CS of the first label. Prototypes (5, 6) and (9, 9) originate exclusively from the CS of the second label. Prototypes (3, 8) and (8, 4) are common in both CSs.

By applying the binary relevance transformation and utilizing CNN on each label separately, MLCNN-BR constructs a final multilabel CS that includes prototypes with the corresponding labels. The merging process ensures that the final CS captures relevant prototypes from each label’s CS.

Table 1. Condensing set for the first label.

Instances	First Label
(1, 1)	1
(1, 8)	1
(2, 7)	0
(3, 8)	1
(4, 5)	1
(7, 1)	0
(8, 4)	1
(9, 1)	1

Table 2. Condensing set for the second label.

Instances	Second Label
(1, 1)	0
(2, 7)	0
(3, 8)	1
(5, 6)	1
(7, 5)	0
(8, 4)	1
(9, 9)	1

Table 3. Final merged condensing set.

Instances	First Label	Second Label
(1, 1)	1	0
(1, 8)	1	0
(3, 8)	1	1
(4, 5)	1	0
(5, 6)	0	1
(8, 4)	1	1
(9, 1)	1	0
(9, 9)	0	1

The MLIB2-BR algorithm shares the same procedure as the MLCNN-BR algorithm, with the only difference being that it performs a single complete iteration on the training set before terminating.

6. Experimental Study

6.1. Experimental Setup

In our experimentation, we utilized nine multilabel datasets provided by Mulan dataset repository [11]. These datasets consisted of numeric features and contained a minimum of five hundred (500) instances. The key characteristics of the used datasets are summarized in Table 4. The last two columns of the table present the dataset cardinality and density. Cardinality represents the average number of labels per instance, while density is calculated by dividing the cardinality by the total number of labels. The domain of each dataset is listed in the second column of Table 4.

Since the datasets encompass features with different value ranges, this can impact the classification process as higher-valued features may dominate the distance calculation between instances. To address this, we normalized the values of all features to the [0,1] range. The normalization was performed using the MinMaxScaler from the scikit-learn Python library [34].

Subsequently, the datasets were split into training and test sets using the stratified 5-fold cross-validation method [35]. This approach ensures that the estimates have reduced variance and improves the generalization performance estimation of classification algorithms. Stratified cross-validation guarantees that the proportion of the feature of interest remains the same in the original data, training set and test set. This ensures that no values are over- or under-represented in the training and test sets, providing a more accurate evaluation of performance and error.

For the implementation of CNN and IB2 and the proposed variations of them, we used Python 3.12.0 and employed Multiprocessing. The utilization of Multiprocessing allows for the concurrent processing of multiple distinct parts of the same Python script by two or more CPU threads. This not only improves processing speed but also enables handling larger volumes of data.

Table 4. Dataset characteristics.

Datasets	Domain	Size	Attributes	Labels	Cardinality	Density
CAL500 (CAL)	Music	502	68	174	26.044	0.150
Emotions (EMT)	Music	593	72	6	1.869	0.311
Water quality (WQ)	Chemistry	1060	16	14	5.073	0.362
Scene (SC)	Image	2407	294	6	1.074	0.179
Yeast (YS)	Biology	2417	103	14	4.237	0.303
Birds (BRD)	Sounds	645	260	19	1.014	0.053
CHD49 (CHD)	Medicine	555	49	6	2.580	0.430
Image (IMG)	Image	2000	294	5	1.236	0.247
Mediamill (MDM)	Video	43,907	120	101	4.376	0.043

The objective of data reduction is to selectively choose training instances to be used as input for instance-based classifiers. This process involves identifying and potentially eliminating redundant instances. The ultimate goal is to obtain smaller datasets that effectively represent the original dataset. The aim is to simplify the dataset, improve computational efficiency and potentially enhance classification performance [36]. It is essential to ensure that the condensing set retains an acceptable amount of information compared to the original dataset.

Therefore, we evaluate the performance of BR*k*NN when it runs over the initial training set and over the condensing sets generated by the proposed algorithms. To measure the effectiveness, we obtained the reduction rate and Hamming loss through a five-fold stratified cross-validation framework. The reduction rate is calculated by comparing the number of instances before and after the reduction process. Thus, a reduction rate of 90% means that 90% of the original training set instances are discarded and the final condensing set consists of 10% of the original instances.

The utilization of eager multilabel classifiers in the experimental study does not align with the objective of the proposed variations, which aim to enhance the speed of instance-based classifiers in multilabel domains. Thus, we do not include eager multilabel classifiers in the experimental study.

As the computational cost of the BR*k*NN classifier is dependent on the size of the training set utilized, our study does not report the CPU time required for classification. Essentially, a higher reduction rate corresponds to a lower computational cost for the BR*k*NN classifier during the classification process. The effectiveness of the predictions is evaluated by computing the Hamming loss, which measures the ratio of incorrectly predicted labels to the total number of labels. The Hamming loss is computed as follows:

$$HL = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \Delta Z_i|}{|L|}$$

Y_i represents the set of actual labels for each instance, while Z_i represents the set of predicted labels for each instance. The total number of instances in the dataset is denoted as m, and $|L|$ refers to the total number of labels. The symmetric difference between two sets, denoted as Δ, can be visualized as the XOR operation. In other words, $|Y_i \Delta Z_i|$ is the number of non-matching labels between the labelsets of the two instances.

For instance, let us consider a multilabel dataset with five labels and a testing instance $x1$ with an actual label set of 11001 and a predicted label set of 11010. In this case, the Hamming loss is calculated as $\frac{2}{5} = 0.4$ because two of the labels do not match (these are fourth and fifth labels). Similarly, if another testing instance $x2$ has an actual label set of 00001 and a predicted label set of 11010, the Hamming loss is $\frac{4}{5} = 0.8$ because all but the third label do not match. To calculate the Hamming loss for a testing set comprising these two instances, we compute the average as $\frac{1}{2} \times (0.4 + 0.8) = 0.6$.

Lastly, it is worth mentioning that, in accordance with established conventions in the relevant literature (e.g., [4,5]), all experiments were conducted using $k = 1$.

6.2. Experimental Results

Table 5 presents the measurements of the experimental study, reporting the Hamming loss (HL) and reduction rates (RR) achieved by MLCNN-H, MLCNN-J, MLCNN-BR, MLCNN-L, MLIB2-H, MLIB2-J, MLIB2-BR and MLIB2-L for each dataset.

MLCNN-H achieved reduction rates ranging from 8.28% to 55.70%, MLCNN-J ($JD > 0.5$) from 0.75% to 51.93%, MLCNN-J ($JD > 0.75$) from 19.45% to 60.62%, MLCNN-BR from 0% to 58.18% and MLCNN-L from 0.10% to 43.05%. MLIB2-H achieved reduction rates ranging from 14.26% to 73.23%, MLIB2-J ($JD > 0.5$) from 1.20% to 73.23%, MLIB2-J ($JD > 0.75$) from 26.28% to 84.67%, MLIB2-BR from 0% to 75.69% and MLIB2-L from 0.35% to 58.85%.

Figure 2 illustrates the reduction rates achieved on each dataset by each algorithm. Instead of the reduction rate, we report the percentage of retained instances for each dataset (in other words, we report 1—*reduction_rate*.

On average, we observe that MLIB2-J ($JD > 0.75$) achieves the highest reduction rate, followed by MLIB2-H, MLCNN-J ($JD > 0.75$) and MLIB2-J ($JD > 0.75$). It is important to note that the distribution of instances within the dataset greatly influence the reduction rate. As the data are complex and not uniformly distributed in space, we observe significant fluctuations in the reduction rate across different datasets. However, in general, we notice that MLCNN-BR and MLIB2-BR exhibit less stable behavior in terms of the reduction rate achieved for each dataset compared to other algorithms.

Additionally, in Table 5, we observe only small differences in Hamming loss between the BR*k*NN classifier that utilizes the initial training set and the BR*k*NN classifier that employs the multilabel condensing sets created by MLCNN-H, MLCNN-J ($JD > 0.5$), MLCNN-J ($JD > 0.75$), MLCNN-BR, MLCNN-L, MLIB2-H, MLIB2-J (0.5), MLIB2-J ($JD > 0.75$), MLIB2-BR and MLIB2-L. In the following section, we provide a statistical analysis to further explore the performance of the algorithms.

Furthermore, as expected, the MLIB2 algorithm achieves higher reduction rates in all its versions than the corresponding versions of MLCNN. This is because IB2 performs one pass on the data compared to the multiple passes performed by CNN, which move more instances in the CS.

The proposed algorithms achieve instance reduction while maintaining high levels of accuracy.

6.3. Statistical Comparisons

In line with the commonly employed approach in the domain of PS and PG algorithms [4,5,37–41], we have supplemented the experimental study with a Wilcoxon signed rank test [42]. This test serves to statistically validate the accuracy of the measurements presented in Table 5. By comparing all the algorithms in pairs based on their performance on each dataset, the Wilcoxon signed rank test confirms their relative rankings. We performed the Wilcoxon signed rank test using the PSPP 2.0.0 statistical software.

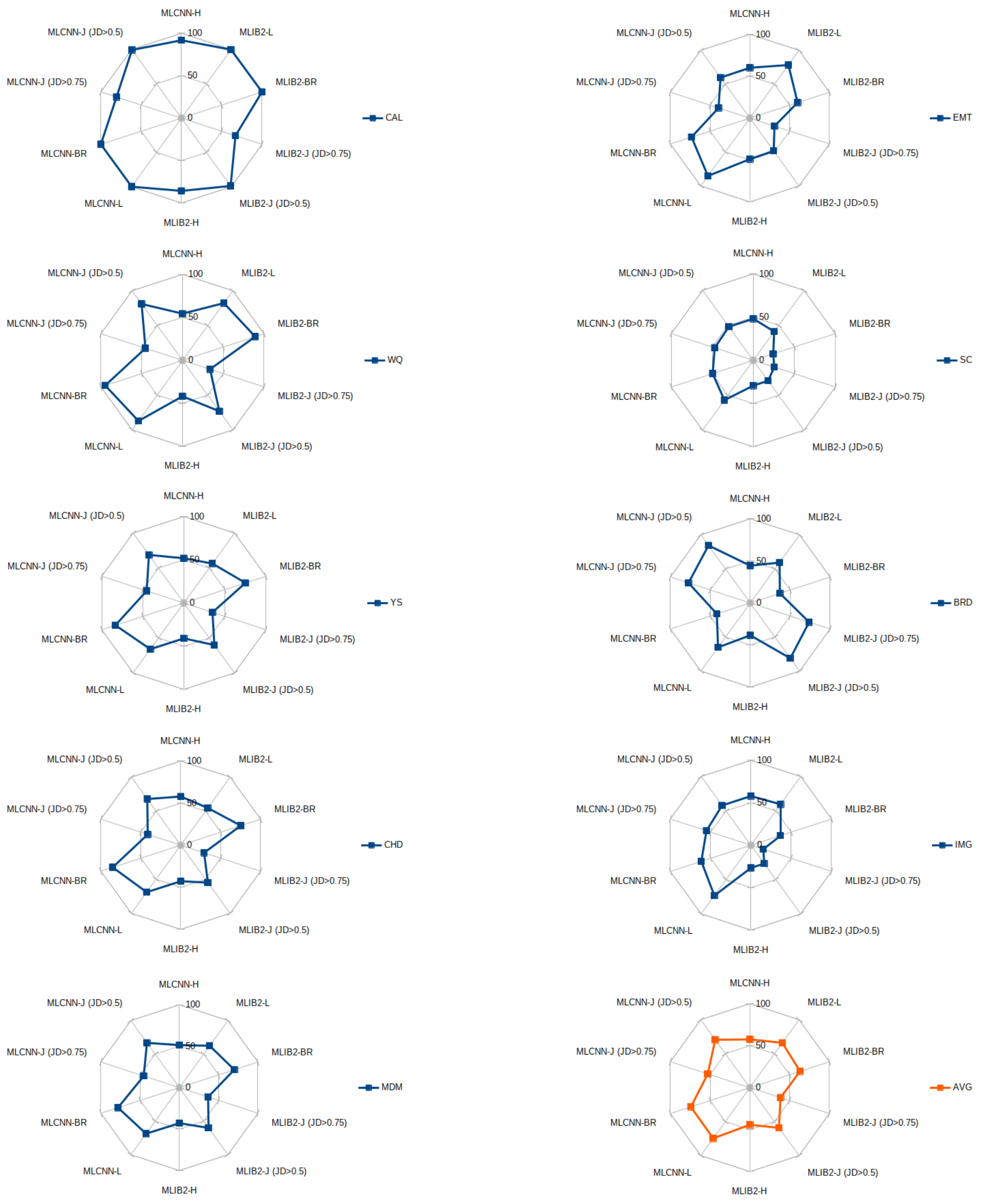

Figure 2. Percentage of retained instances per dataset (CAL, EMT, WQ, SC, YS, BRD, CHD, IMG, MDM) and algorithm. Last figure reports the average percentage of retained instances over all datasets and for each algorithm.

Table 5. Comparison table of the reduction rate (RR (%)) and the Hamming loss (HL (%)).

Dataset		BR*k*NN	MLCNN-H	MLCNN-J ($JD > 0.5$)	MLCNN-J ($JD > 0.75$)	MLCNN-BR	MLCNN-L	MLIB2-H	MLIB2-J ($JD > 0.5$)	MLIB2-J ($JD > 0.75$)	MLIB2-BR	MLIB2-L
CAL	HL:	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.18	0.19
	RR:	-	8.28	0.75	19.45	0.0	0.10	14.26	1.20	32.92	0.0	0.35
EMT	HL:	0.24	0.26	0.26	0.25	0.29	0.24	0.26	0.26	0.26	0.30	0.24
	RR:	-	39.88	40.26	60.62	26.64	14.67	51.05	51.64	69.06	39.97	21.67
WQ	HL:	0.33	0.34	0.33	0.34	0.36	0.33	0.35	0.33	0.34	0.37	0.33
	RR:	-	45.71	18.99	54.65	5.33	12.74	58.16	26.68	66.20	10.85	17.73
SC	HL:	0.11	0.12	0.12	0.12	0.13	0.12	0.14	0.14	0.14	0.17	0.14
	RR:	-	51.93	51.93	53.03	50.44	43.05	70.75	70.75	74.50	75.69	58.85
YS	HL:	0.24	0.27	0.26	0.26	0.30	0.26	0.27	0.26	0.27	0.30	0.26
	RR:	-	48.26	31.27	54.72	16.61	33.95	59.09	39.80	65.07	24.90	43.61
BRD	HL:	0.05	0.06	0.05	0.05	0.08	0.05	0.06	0.05	0.05	0.08	0.05
	RR:	-	55.70	15.50	22.71	58.18	34.81	61.59	18.80	26.28	62.87	40.62
CHD	HL:	0.35	0.36	0.36	0.38	0.40	0.37	0.37	0.37	0.38	0.40	0.38
	RR:	-	42.18	32.20	58.64	14.74	30.98	57.20	44.97	70.82	24.89	45.33
IMG	HL:	0.20	0.22	0.22	0.21	0.23	0.21	0.25	0.25	0.25	0.26	0.22
	RR:	-	42.11	42.11	44.82	38.43	26.61	73.23	73.23	84.67	63.43	40.47
MDM	HL:	0.031	0.035	0.032	0.036	0.042	0.032	0.037	0.033	0.039	0.043	0.033
	RR:	-	48.70	33.19	54.58	21.76	31.30	57.08	39.90	63.43	30.06	37.61

We have also utilized the non-parametric Friedman test to rank the algorithms individually for each dataset. This test assigned a rank to each algorithm, with the best performer receiving rank 1, the second best receiving rank 2 and so on. The PSPP statistical software was employed to conduct the Friedman test that was executed twice, once for each measured criterion.

6.3.1. Wilcoxon Signed Rank Test Results

The results of the Wilcoxon signed rank test for the Hamming loss (ACC) and Reduction Rate (RR) measurements are shown in Table 6. The column labeled "w/l/t" presents the number of wins, losses and ties for each comparison test. The last column, labeled "Wilc.", indicates a numerical value that quantifies the significance of the difference between the two compared algorithms. If this value is less than 0.05, it can be concluded that the difference is statistically significant. In Table 6, the Wilc. values that is less than 0.05 are in bold face.

The results indicate that there is no statistical difference in accuracy between the BR*k*NN classifier and the proposed MLCNN-L and MLIB2-L. Therefore, the test proves that the BR*k*NN classifier that operates on the CS generated by the proposed MLCNN-L and MLIB2-L algorithms achieves a comparable level of accuracy to the conventional BR*k*NN classifier that operates on the original training set.

Moreover, the test shows that there is a significant statistical difference in accuracy between BR*k*NN and the remaining multilabel variations in CNN. Nevertheless, in certain datasets, MLCNN-J ($JD > 0.5$) and MLIB2-J ($JD > 0.5$) achieve a comparable level of accuracy to the "conventional" BR*k*NN classifier that operates on the original training set. Moreover, MLCNN-L achieved the best Hamming loss in comparison to all other algorithms and close to the one of BR*k*NN.

Furthermore, the test affirms that there is statistical difference in terms of accuracy between the pairs MLCNN-L versus MLIB2-BR, MLIB2-J ($JD > 0.5$), MLIB2-L, MLCNN-H, MLCNN-BR, MLIB2-H, MLIB2-J ($JD > 0.75$) and MLCNN-J ($JD > 0.75$). The version of MLCNN-J ($JD > 0.5$) presents discrepancy in statistical terms of accuracy with the versions MLCNN-H, MLCNN-BR, MLIB2-H, MLIB2-J ($JD > 0.75$) and MLIB2-BR.

According to the Wilcoxon test, we have statistical difference in terms of reduction rate between the pairs MLIB2-H and the versions MLCNN-BR, MLCNN-H, MLIB2-J ($JD > 0.5$), MLIB2-BR, MLCNN-L, MLIB2-L and MLCNN-J ($JD > 0.5$). Moreover, the pairs between MLIB2-J ($JD > 0.75$) and the versions MLIB2-J ($JD > 0.5$), MLIB2-BR, MLCNN-L and MLIB2-L have statistical difference in terms of reduction rate.

Table 6. Results of Wilcoxon signed rank test on ACC and RR measurements.

Methods	**Accuracy**		**Reduction Rate**	
	w/l/t	**Wilc.**	**w/l/t**	**Wilc.**
BR*k*NN vs. MLCNN-H	8/0/1	**0.012**	-	-
BR*k*NN vs. MLCNN-J ($JD > 0.5$)	6/0/3	**0.027**	-	-
BR*k*NN vs. MLCNN-J ($JD > 0.75$)	7/0/2	**0.018**	-	-
BR*k*NN vs. MLCNN-BR	8/0/1	**0.012**	-	-
BR*k*NN vs. MLIB2-H	8/0/1	**0.012**	-	-
BR*k*NN vs. MLIB2-J ($JD > 0.5$)	6/0/3	**0.026**	-	-
BR*k*NN vs. MLIB2-J ($JD > 0.75$)	7/0/2	**0.018**	-	-
BR*k*NN vs. MLIB2-BR	8/1/0	**0.011**	-	-
BR*k*NN vs. MLCNN-L	5/1/3	**0.046**	-	-
BR*k*NN vs. MLIB2-L	5/1/3	**0.046**	-	-
MLCNN-H vs. MLCNN-J ($JD > 0.5$)	1/6/2	**0.028**	6/1/2	**0.028**
MLCNN-H vs. MLCNN-J ($JD > 0.75$)	3/6/0	0.260	1/8/0	0.110
MLCNN-H vs. MLCNN-BR	1/8/0	**0.011**	8/1/0	**0.015**

Table 6. *Cont.*

Methods	Accuracy		Reduction Rate	
	w/l/t	**Wilc.**	**w/l/t**	**Wilc.**
MLCNN-H vs. MLIB2-H	8/1/0	**0.011**	0/9/0	**0.008**
MLCNN-H vs. MLIB2-J ($JD > 0.5$)	4/5/0	0.859	5/4/0	0.767
MLCNN-H vs. MLIB2-J ($JD > 0.75$)	7/2/0	0.066	1/8/0	0.086
MLCNN-H vs. MLIB2-BR	8/1/0	**0.015**	5/4/0	0.515
MLCNN-H vs. MLCNN-L	1/8/0	**0.021**	9/0/0	**0.008**
MLCNN-H vs. MLIB2-L	2/7/0	0.260	7/2/0	0.051
MLCNN-J ($JD > 0.5$) vs. MLCNN-J ($JD > 0.75$)	6/3/0	0.214	0/9/0	**0.008**
MLCNN-J ($JD > 0.5$) vs. MLCNN-BR	8/1/0	**0.011**	8/1/0	0.110
MLCNN-J ($JD > 0.5$) vs. MLIB2-H	9/0/0	**0.008**	0/9/0	**0.008**
MLCNN-J ($JD > 0.5$) vs. MLIB2-J ($JD > 0.5$)	8/1/0	**0.011**	0/9/0	**0.008**
MLCNN-J ($JD > 0.5$) vs. MLIB2-J ($JD > 0.75$)	9/0/0	**0.008**	0/9/0	**0.008**
MLCNN-J ($JD > 0.5$) vs. MLIB2-BR	8/1/0	**0.011**	6/3/0	0.859
MLCNN-J ($JD > 0.5$) vs. MLCNN-L	1/8/0	0.051	7/2/0	0.214
MLCNN-J ($JD > 0.5$) vs. MLIB2-L	3/6/0	0.678	4/5/0	0.314
MLCNN-J ($JD > 0.75$) vs. MLCNN-BR	8/1/0	**0.011**	8/1/0	0.051
MLCNN-J ($JD > 0.75$) vs. MLIB2-H	7/2/0	**0.066**	3/6/0	0.214
MLCNN-J ($JD > 0.75$) vs. MLIB2-J ($JD > 0.5$)	3/6/0	0.953	7/2/0	0.374
MLCNN-J ($JD > 0.75$) vs. MLIB2-J ($JD > 0.75$)	7/2/0	**0.021**	0/9/0	**0.008**
MLCNN-J ($JD > 0.75$) vs. MLIB2-BR	8/1/0	**0.015**	6/3/0	0.260
MLCNN-J ($JD > 0.75$) vs. MLCNN-L	0/9/0	**0.008**	8/1/0	**0.015**
MLCNN-J ($JD > 0.75$) vs. MLIB2-L	2/7/0	0.214	7/2/0	0.086
MLCNN-BR vs. MLIB2-H	3/6/0	0.139	0/9/0	**0.008**
MLCNN-BR vs. MLIB2-J ($JD > 0.5$)	3/6/0	0.066	1/8/0	0.110
MLCNN-BR vs. MLIB2-J ($JD > 0.75$)	3/6/0	0.173	1/8/0	**0.015**
MLCNN-BR vs. MLIB2-BR	8/1/0	**0.038**	0/8/1	**0.012**
MLCNN-BR vs. MLCNN-L	1/8/0	**0.011**	4/5/0	0.953
MLCNN-BR vs. MLIB2-L	2/7/0	**0.021**	2/7/0	0.139
MLIB2-H vs. MLIB2-J ($JD > 0.5$)	1/6/2	0.056	6/1/2	**0.028**
MLIB2-H vs. MLIB2-J ($JD > 0.75$)	5/4/0	1.000	1/8/0	0.110
MLIB2-H vs. MLIB2-BR	8/1/0	**0.015**	7/2/0	**0.021**
MLIB2-H vs. MLCNN-L	0/9/0	**0.008**	9/0/0	**0.008**
MLIB2-H vs. MLIB2-L	1/8/0	**0.028**	9/0/0	**0.008**
MLIB2-J ($JD > 0.5$) vs. MLIB2-J ($JD > 0.75$)	7/2/0	0.051	0/9/0	**0.008**
MLIB2-J ($JD > 0.5$) vs. MLIB2-BR	8/1/0	**0.011**	7/2/0	0.173
MLIB2-J ($JD > 0.5$) vs. MLCNN-L	1/8/0	**0.011**	8/1/0	0.051
MLIB2-J ($JD > 0.5$) vs. MLIB2-L	1/7/1	0.093	6/3/0	0.214
MLIB2-J ($JD > 0.75$) vs. MLIB2-BR	8/1/0	**0.015**	7/2/0	**0.066**
MLIB2-J ($JD > 0.75$) vs. MLCNN-L	0/9/0	**0.008**	8/1/0	**0.011**
MLIB2-J ($JD > 0.75$) vs. MLIB2-L	0/9/0	**0.008**	8/1/0	**0.011**
MLIB2-BR vs. MLCNN-L	1/8/0	**0.011**	4/5/0	0.374
MLIB2-BR vs. MLIB2-L	1/8/0	**0.011**	4/5/0	0.678
MLCNN-L vs. MLIB2-L	4/1/4	0.078	0/9/0	**0.008**

6.3.2. Friedman Test Results

The results of the Friedman test for the ACC and RR measurements are displayed in Table 7. Notice that the RR ranks are inverted; i.e., the larger the number, the higher the rank of the algorithm is. The Friedman test shows that

Table 7. Results of Friedman test on ACC and RR measurements.

Algorithm	Mean Rank	
	ACC	**RR**
MLCNN-H	6.00	5.78
MLCNN-J ($JD > 0.5$)	4.22	4.00
MLCNN-J ($JD > 0.75$)	5.17	7.22
MLCNN-BR	9.00	2.28
MLCNN-L	3.67	2.67
MLIB2-H	7.72	8.33
MLIB2-J ($JD > 0.5$)	5.72	6.11
MLIB2-J ($JD > 0.75$)	7.44	9.22
MLIB2-BR	9.72	4.61
MLIB2-L	5.06	4.78
BR*k*NN	2.28	-

- MLCNN-L is the most accurate approach. MLCNN-J ($JD > 0.5$), MLCNN-J ($JD > 0.75$) and MLIB2-L are the runners up.
- MLIB2-J ($JD > 0.75$) and MLIB2-H achieve the highest reduction rates. MLCNN-J ($JD > 0.75$) and MLIB2-J ($JD > 0.5$) are the runners up.

7. Conclusions

The main objective of this paper is to address the issue of data reduction techniques specifically tailored for multilabel datasets. Here, the focus is on reducing instances rather than features. This type of reduction is crucial in the context of instance-based classification as it helps mitigate the computational burden associated with large datasets. However, it is important to note that most existing data reduction techniques are primarily designed for single-label classification problems and are not well-suited for multilabel classification. Additionally, these techniques cannot seamlessly integrate with problem transformation methods such as binary relevance or label powerset, which are commonly used in multilabel classification scenarios.

This paper presents novel algorithms focused on accelerating the instance-based classifiers in the context of multilabel classification. The study introduces four variations of the well-known CNN-rule and four variations of IB2 specifically designed for multilabel classification. The proposed MLCNN-H, MLCNN-J, MLCNN-L and MLCNN-BR algorithms and the corresponding MLIB2-H, MLIB2-J, MLIB2-L and MLIB2-BR algorithms can be considered as the first prototype selection algorithms for multilabel data condensing.

The proposed algorithms do not require any specific parameters. MLCNN-H and MLIB2-H consider two neighboring instances to be different if their Hamming distance exceeds the dataset density. MLCNN-J and MLIB2-J identify two neighboring instances as distinct if their Jaccard distance surpasses a predefined threshold. MLCNN-BR and MLIB2-BR construct separate prototypes for each label using the conventional CNN method and subsequently merge them by combining different labels to form the final condensing sets. Finally, in MLCNN-L and MLIB2-L, if the Levenshtein distance between two neighboring instances exceeds half the cardinality of the dataset, they are considered as different.

Consequently, the proposed algorithms generate a multilabel condensing set. This condensing set can be utilized by BR*k*NN to conduct multilabel prediction.

The experimental study demonstrated that switching from the initial training set to the condensing sets produced by the proposed algorithms did not greatly affect the accuracy achieved by BR*k*NN. However, it reduced the computational cost required for the classification process. The new variations achieved a reduction of more than 50% in computational costs.

Looking in more detail at the results of the experiments, it appears that, in terms of the overall classification performance of the algorithms compared to BR*k*NN, the MLCNN-L and MLCNN-J ($JD > 0.5$) outperformed the other variations. Further, regarding the algorithms' overall reduction rate performance, the MLIB2-J ($JD > 0.75$) and MLIB2-H achieved superior results.

This study highlights the ongoing significance of data reduction in multilabel problems within the domains of data mining and machine learning. Our goal is to extend popular data reduction techniques, typically applied to single-label datasets, to the realm of multilabel datasets. Additionally, our future work involves the development of novel parameter-free data reduction methods, as well as scalable approaches for training set classification in the context of multilabel problems.

Author Contributions: Conceptualisation, P.F., S.O. and G.E.; methodology, P.F., S.O. and G.E.; software, P.F., S.O. and G.E.; validation, P.F., S.O. and G.E.; formal analysis, P.F., S.O. and G.E.; investigation, P.F., S.O. and G.E.; resources, P.F., S.O. and G.E.; data curation, P.F., S.O. and G.E.; writing—original draft preparation, P.F., S.O. and G.E.; writing—review and editing, P.F., S.O. and G.E.; visualisation, P.F., S.O. and G.E.; supervision, S.O. and G.E.; project administration, P.F., S.O. and G.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. These data can be found here: https://mulan.sourceforge.net/datasets-mlc.html (accessed on 17 October 2023) and https://www.uco.es/kdis/mllresources/ (accessed on 17 October 2023).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DRT	Data Reduction Technique
PS	Prototype Selection
CNN	Condensed Nearest Neighbor
IB2	Instance-Based Learning 2
CS	Condensing set
TS	Training set
MLCNN-H	Multilabel Condensed Nearest Neighbor with Hamming Distance
MLCNN-J	Multilabel Condensed Nearest Neighbor with Jaccard Distance
MLCNN-L	Multilabel Condensed Nearest Neighbor with Levenshtein Distance
MLCNN-BR	Multilabel Condensed Nearest Neighbor with Binary Relevance
MLIB2-H	Multilabel Instance-Based Learning 2 with Hamming Distance
MLIB2-J	Multilabel Instance-Based Learning 2 with Jaccard Distance
MLIB2-L	Multilabel Instance-Based Learning 2 with Levenshtein Distance
MLIB2-BR	Multilabel Instance-Based Learning 2 with Binary Relevance

References

1. Tsoumakas, G.; Katakis, I. Multi-label classification: An overview. *Int. J. Data Warehous. Min.* **2007**, *3*, 1–13. [CrossRef]
2. Cover, T.; Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27. [CrossRef]
3. Liu, H.; Motoda, H. *Feature Selection for Knowledge Discovery and Data Mining*; Kluwer Academic Publishers: New York, NY, USA, 1998.
4. Garcia, S.; Derrac, J.; Cano, J.; Herrera, F. Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 417–435. [CrossRef] [PubMed]
5. Triguero, I.; Derrac, J.; Garcia, S.; Herrera, F. A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification. *Trans. Systems Man Cyber Part C* **2012**, *42*, 86–100. [CrossRef]

6. Spyromitros, E.; Tsoumakas, G.; Vlahavas, I. An Empirical Study of Lazy Multilabel Classification Algorithms. In *Proceedings of the Artificial Intelligence: Theories, Models and Applications*; Darzentas, J.; Vouros, G.A.; Vosinakis, S.; Arnellos, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 401–406. [CrossRef]
7. Hart, P.E. The condensed nearest neighbor rule. *IEEE Trans. Inf. Theory* **1967**, *18*, 515–516.
8. Aha, D.W.; Kibler, D.; Albert, M.K. Instance-based learning algorithms. *Mach. Learn.* **1991**, *6*, 37–66. [CrossRef]
9. Filippakis, P.; Ougiaroglou, S.; Evangelidis, G. Condensed Nearest Neighbour Rules for Multi-Label Datasets. In Proceedings of the International Database Engineered Applications Symposium Conference, Heraklion, Greece, 5–7 May 2023; pp. 43–50. [CrossRef]
10. Levenshtein, V.I. Binary codes capable of correcting deletions, insertions, and reversals. *Sov. Phys. Dokl.* **1965**, *10*, 707–710.
11. Tsoumakas, G.; Spyromitros-Xioufis, E.; Vilcek, J.; Vlahavas, I. Mulan: A Java Library for Multi-Label Learning. *J. Mach. Learn. Res.* **2011**, *12*, 2411–2414.
12. Read, J.; Reutemann, P.; Pfahringer, B.; Holmes, G. MEKA: A Multi-label/Multi-target Extension to WEKA. *J. Mach. Learn. Res.* **2016**, *17*, 1–5.
13. Charte, F.; Rivera, A.J.; del Jesus, M.J.; Herrera, F. MLeNN: A First Approach to Heuristic Multilabel Undersampling. In *Intelligent Data Engineering and Automated Learning–IDEAL 2014*; Springer: New York, NY, USA, 2014; pp. 1–9. [CrossRef]
14. Wilson, D.L. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Trans. Syst. Man Cybern.* **1972**, *SMC-2*, 408–421. [CrossRef]
15. Kanj, S.; Abdallah, F.; Denœux, T.; Tout, K. Editing training data for multi-label classification with the k-nearest neighbor rule. *Pattern Anal. Appl.* **2015**, *19*, 145–161. [CrossRef]
16. Arnaiz-González, Á; Díez-Pastor, J.F.; Rodríguez, J.J.; García-Osorio, C. Local sets for multi-label instance selection. *Appl. Soft Comput.* **2018**, *68*, 651–666. [CrossRef]
17. Leyva, E.; González, A.; Pérez, R. Three new instance selection methods based on local sets: A comparative study with several approaches from a bi-objective perspective. *Pattern Recognit.* **2015**, *48*, 1523–1537. [CrossRef]
18. Li, H.; Fang, M.; Li, H.; Wang, P. Prototype selection for multi-label data based on label correlation. *Neural Comput. Appl.* **2023**. [CrossRef]
19. Chou, C.H.; Kuo, B.H.; Chang, F. The Generalized Condensed Nearest Neighbor Rule as A Data Reduction Method. In Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06), Hong Kong, China, 20–24 August 2006; Volume 2, pp. 556–559. [CrossRef]
20. Suyal, H.; Singh, A. Improving Multi-Label Classification in Prototype Selection Scenario. In *Computational Intelligence and Healthcare Informatics*; Wiley: Hoboken, NJ, USA, 2021; pp. 103–119. [CrossRef]
21. Zhang, M.L.; Zhou, Z.H. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognit.* **2007**, *40*, 2038–2048. [CrossRef]
22. Arnaiz-González, Á.; Díez-Pastor, J.F.; Rodríguez, J.J.; García-Osorio, C. Study of data transformation techniques for adapting single-label prototype selection algorithms to multi-label learning. *Expert Syst. Appl.* **2018**, *109*, 114–130. [CrossRef]
23. Calvo-Zaragoza, J.; Valero-Mas, J.J.; Rico-Juan, J.R. Improving kNN multi-label classification in Prototype Selection scenarios using class proposals. *Pattern Recognit.* **2015**, *48*, 1608–1622. [CrossRef]
24. González, M.; Cano, J.R.; García, S. ProLSFEO-LDL: Prototype Selection and Label- Specific Feature Evolutionary Optimization for Label Distribution Learning. *Appl. Sci.* **2020**, *10*, 3089. [CrossRef]
25. Geng, X. Label Distribution Learning. *IEEE Trans. Knowl. Data Eng.* **2016**, *28*, 1734–1748. [CrossRef]
26. Ougiaroglou, S.; Filippakis, P.; Evangelidis, G. Prototype Generation for Multi-label Nearest Neighbours Classification. In *Proceedings of the Hybrid Artificial Intelligent Systems*; Sanjurjo González, H., Pastor López, I., García Bringas, P., Quintián, H., Corchado, E., Eds.; Springer: Cham, Germany, 2021; pp. 172–183.
27. Ougiaroglou, S.; Filippakis, P.; Fotiadou, G.; Evangelidis, G. Data reduction via multi-label prototype generation. *Neurocomputing* **2023**, *526*, 1–8. [CrossRef]
28. Sánchez, J. High training set size reduction by space partitioning and prototype abstraction. *Pattern Recognit.* **2004**, *37*, 1561–1564. [CrossRef]
29. Valero-Mas, J.J.; Gallego, A.J.; Alonso-Jiménez, P.; Serra, X. Multilabel Prototype Generation for data reduction in K-Nearest Neighbour classification. *Pattern Recognit.* **2023**, *135*, 109190. [CrossRef]
30. Chen, C.; Jóźwik, A. A sample set condensation algorithm for the class sensitive artificial neural network. *Pattern Recognit. Lett.* **1996**, *17*, 819–823. [CrossRef]
31. Sun, L.; Ji, S.; Ye, J. Hypergraph Spectral Learning for Multi-Label Classification. In Proceedings of the Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; pp. 668–676. [CrossRef]
32. Byerly, A.; Kalganova, T. Class Density and Dataset Quality in High-Dimensional, Unstructured Data. *arXiv* **2022**. [CrossRef]
33. Zhang, S.; Hu, Y.; Bian, G. Research on string similarity algorithm based on Levenshtein Distance. In Proceedings of the 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 25–26 March 2017; pp. 2247–2251. [CrossRef]
34. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.

35. Sechidis, K.; Tsoumakas, G.; Vlahavas, I. On the Stratification of Multi-label Data. In *Proceedings of the Machine Learning and Knowledge Discovery in Databases*; Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 145–158. [CrossRef]
36. Czarnowski, I.; Jędrzejowicz, P. An Approach to Data Reduction for Learning from Big Datasets: Integrating Stacking, Rotation, and Agent Population Learning Techniques. *Complexity* **2018**, *2018*, 7404627. [CrossRef]
37. Gallego, A.J.; Calvo-Zaragoza, J.; Valero-Mas, J.J.; Rico-Juan, J.R. Clustering-Based k-Nearest Neighbor Classification for Large-Scale Data with Neural Codes Representation. *Pattern Recogn.* **2018**, *74*, 531–543. [CrossRef]
38. Ougiaroglou, S.; Evangelidis, G. RHC: Non-Parametric Cluster-Based Data Reduction for Efficient k-NN Classification. *Pattern Anal. Appl.* **2016**, *19*, 93–109. [CrossRef]
39. Escalante, H.J.; Graff, M.; Morales-Reyes, A. PGGP: Prototype Generation via Genetic Programming. *Appl. Soft Comput.* **2016**, *40*, 569–580. [CrossRef]
40. Escalante, H.J.; Marin-Castro, M.; Morales-Reyes, A.; Graff, M.; Rosales-Pérez, A.; Montes-Y-Gómez, M.; Reyes, C.A.; Gonzalez, J.A. MOPG: A Multi-Objective Evolutionary Algorithm for Prototype Generation. *Pattern Anal. Appl.* **2017**, *20*, 33–47. [CrossRef]
41. Calvo-Zaragoza, J.; Valero-Mas, J.J.; Rico-Juan, J.R. Prototype Generation on Structural Data Using Dissimilarity Space Representation. *Neural Comput. Appl.* **2017**, *28*, 2415–2424. [CrossRef]
42. Sheskin, D. *Handbook of Parametric and Nonparametric Statistical Procedures*; A Chapman & Hall Book; Chapman & Hall/CRC: Boca Raton, FL, USA, 2011.