A Fast Density Based Outlier Aware Supervised Approach for Instance Selection

Danial Chakma and Md. Shariful Islam
Computer Science and Engineering Department,
Bangladesh University of Engineering and Technology,
Dhaka, Bangladesh
Email: danial08cse@gmail.com, sharif@gmail.com

***Abstract*—In modern Era of data science and data mining, increased volume of both unstructured and structured data poses a great challenge in the field of data mining, which limit our capabilities of converting this increased volume data into insightful knowledge. In order to deal with this increased size of datasets, various techniques for instance selection have been proposed to reduce the dataset to a manageable volume and, consequently, to reduce the computational resources that are necessary to apply data mining approaches. However, most of the proposed approaches for instance selection have a high time complexity and, due to this, they cannot be applied for dealing with big data. In this paper, we propose a novel approach for instance selection called LSH-IS. This approach adopts the notion of local density for selecting the most representative instances of each class of the dataset, providing a reasonably low time complexity. The approach was evaluated on some well-known datasets used in a classification task, and its performance was compared to state-of-the-art algorithms in literature, considering three measures: accuracy, reduction, and effectiveness. All the obtained results show that, in general, the LSH-IS algorithm provides the best trade-off between accuracy and reduction.**

I. Introduction

Recent past years have witnessed a dramatic rise to collect a huge volume of data from different sensor devices in different formats in an unprecedented rates. These large collections of data from different devices are usually called **Big Data**, and due to their large size and complexity, they challenge even the capabilities of our current data mining approaches [1]. To address these challenges, instance selection has been considered as an alternative to overcome the challenges raised by the size of these datasets.

Instance selection (IS) is a pre-processing task of data-mining or machine learning that aims to choose a representative subset of instances among the total available data, which allows to carry out a data mining task with no performance loss or, at least, a reduced performance loss [2]. Thus, every IS strategy faces a trade-off between the reduction rate of the dataset and the resulting classification quality [3]. Besides its capability of reducing a big dataset to a manageable subset, instance selection can also be used for improving the learned models through the deletion of useless (redundant), erroneous or noisy instances, before applying learning algorithms [2, 4].

Most of the proposed algorithms for instance selection, such as [5, 6, 7, 8, 9, 10], are designed for preserving the boundaries between different classes in the dataset. This is a reasonable strategy, because border instances provide relevant information for supporting discrimination between classes [6]. However, in general, these algorithms have a high time complexity, and this is not a desirable property for algorithms that should deal with Big Data. The high time complexity of these algorithms results from the fact that they usually should search for the border instances within the whole dataset and, due to this, they usually need to perform comparisons between each pair of instances in the dataset.

In this paper, based on the works of [11] **XLIDS**, we propose an upgrade algorithm for instance selection, **LSH-IS**, which analyses the instances of each class separately and focuses on keeping only the most representative instance in a given (arbitrary) neighborhood then a class independent post processing step which romoves out-liars or noisy data points. Due to this strategy, the resulting time complexity of LSH-IS is reasonably low and twice the run-time of XLIDS. Besides that, it is important to notice that LSH-IS assumes that the representativeness of a given instance x is proportional to its local density ordering, which, intuitively, determines how many instances are less dense than x, within the class of x. In this work, we basically solve the limitations that XLIDS [11] lacking. **Firstly**, XLIDS fails to detect

out-liar or noisy data points when dataset contains noisy data points by the inherent nature of the algorithm. **Secondly**, XLIDS is not suitable for large scale dataset (magnitude of millions) pre-processing when our computing resources (RAMs, CPUs) are limited.

Our approach was evaluated on some well-known datasets and its performance was compared with the performance of important algorithms provided by the literature, according to 3 different performance measures: accuracy, reduction and effectiveness (proposed in [12]). The accuracy was evaluated considering the KNN algorithm [13]. The results show that, compared to the other algorithms, XLDIS provides the best trade-off between accuracy and reduction, while presents a reasonably low time complexity. Section II presents some related works. Section III presents the notation that will be used throughout the paper. Section IV presents our approach. Section V discusses our experimental evaluation. Finally, Sect. VI presents our main conclusions and remarks.

II. Related Works

In this section, we discuss some important instance reduction methods. In this discussion, we consider T as the original set of instances in the training set and S, with $S \subseteq T$, as the reduced set of instances, resulting from the instance selection process. The Condensed Nearest Neighbor (CNN) algorithm, introduced by [14], randomly selects one instance that belongs to each class from T and puts them in S. Then, each instance $\in$ T is classified using only the instances in S. If an instance is misclassified, it is added to S, in order to ensure that it will be classified correctly. This process is repeated until there is no instance in T that is misclassified. CNN can assign noisy and outlier instances to S, causing harmful effects in the classification accuracy. Also, CNN is dependent on instance order in the training set T . The time complexity of CNN is $O(|T|^2)$, where $|T|$ is the size of the training set. The Reduced Nearest Neighbor algorithm (RNN) [15], assigns all instances in T to S first. After, it removes each instance from S, until further removal causes no other instances in T to be misclassified by the remaining instances in S. RNN is less sensitive to noise than CNN and produces subsets S that are smaller than the subsets produced by CNN. However, the main drawback of RNN is its cubic time complexity, which is higher than the time complexity of CNN.

The Generalized Condensed Nearest Neighbor (GCNN) was proposed by [3]. Considering $d_N(x)$ as the distance between x and its nearest neighbor, and $d_E(x)$ as the distance between x and its nearest enemy (instance of a class that is different from the class of x), x is included by GCNN in S if $d_N(x) - d_E(x) > \rho$, where ρ is an arbitrary threshold. In general, GCNN produces sets S that are smaller than the sets produced by CNN. The main negative point regarding GCNN is that determining the value of ρ can be a challenge.

The Edited Nearest Neighbor (ENN) algorithm [10] assigns all training instances to S first. Then, each instance in S is removed if it does not agree with the label of the majority of its k nearest neighbors. This strategy is effective for improving the classification accuracy of the learned models, because it removes noisy and outlier instances. However, since it keeps internal instances, it cannot reduce the dataset as much as other reduction algorithms. The literature provides some extensions of this method, such as [16].

In [9], the authors present 5 approaches, named the Decremental Reduction Optimization Procedure (DROP). These algorithms assume that each instance x has k nearest neighbors ($k \in N$), and those instances which have x as one of their k nearest neighbors are called the associates of x. Among the proposed algorithms, DROP3 has the best trade-off between the reduction of the dataset and the accuracy of the classification. As an initial step, it applies a noise-filter algorithm such as ENN. Then, it removes an instance x if its associates in the original training set can be correctly classified without x. The main drawback of DROP3 is its high time complexity.

The Iterative Case Filtering algorithm (ICF) [5] is based on the notions of Coverage set and Reachable set. The coverage set of an instance x is the set of instances in T whose distance from x is less than the distance between x and its nearest enemy (instance with a different class). This notion is analogous to the notion of local set, which we adopt in our algorithm. The Reachable set of an instance x, on the other hand, is the set of instances in T that have x in their respective coverage sets. In this method, a given instance x is removed from S if $|Reachable(x)| > |Coverage(x)|$, i.e. when the number of other instances that can classify x correctly is greater than the number of instances that x can correctly classify.

In [6], the authors adopted the notion of local sets for designing three complementary methods for instance selection. In this context, the local set of a given instance x is the set of instances contained in the largest hypersphere centered on x such that it does not contain instances from any other class. The first algorithm, called Local Set-based Smoother (LSSm), was proposed for

removing instances that are harmful for the classification accuracy, i.e. instances that misclassify more instances than those that they correctly classify. It uses two notions for guiding the process: usefulness and harmfulness. The usefulness u(x) of a given instance x is the number of instances having x among the members of their local sets, and the harmfulness $h(x)$ is the number of instances having x as the nearest enemy. For each instance x in T, the algorithm includes x in S if $u(x) \geq h(x)$. Since the primary goal of LSSm is to remove harmful instances, its reduction rate is lower than most of the instance selection algorithms. The second algorithm, called Local Set-based Centroids Selector method (LSCo), firstly applies LSSm to remove noise and then applies LS-clustering [17] to identify clusters in T. The algorithm keeps in S only the centroids of the resulting clusters. Finally, the Local Set Border selector (LSBo) first uses LSSm to remove noise, and then, it computes the local set of every instance ∈ T. Next, the instances in T are sorted in the ascending order of the cardinality of their local sets. In the last step, LSBo verifies, for each instance $x \in T$ if any member of its local set is contained in S, thus ensuring the proper classification of x. If that is not the case, x is included in S to ensure its correct classification. The time complexity of the three approaches is $O(|T|^2)$. Among the three algorithms, LSBo provides the best balance between reduction and accuracy. In [12], the authors proposed the Local Density-based Instance Selection (LDIS) algorithm. This algorithm selects the instances with the highest density values in their neighborhoods. The LDIS algorithm searches for representative instances only among the instances of each class (separately). For this reason, it is not necessary to perform a global search in the whole dataset. As a consequence, the time complexity of the LDIS algorithm is lower than the time complexity of most of the instance selection algorithms. Besides that, this surprisingly simple algorithm is able to produce representative subsets of data that results in high accuracies in classification tasks. The XLDIS algorithm, proposed in this paper, can be viewed as an extension of the LDIS algorithm. Other approaches can be found in surveys such as [2, 18].

III. Notations

In this section, we introduce a notation adapted from [11] that will be used throughout the paper.

- $T = x_1, x_2, ..., x_n$ is the non-empty set of n instances (or data objects), representing the original dataset to be reduced in the instance selection process.
- Each $x_i \in T$ is an m-tuple, such that $x_i = (x_{i1}, x_{i2}, ..., x_{im})$, where x_{ij} represents the value of the j^{th} feature of the instance x_i, for $1 \leq j \leq m$.
- $L = l_1, l_2, ..., l_p$ is the set of p class labels that are used for classifying the instances in T , where each $l_i \in L$ represents a given class label.
- $l : T \mapsto L$ is a function that maps a given instance $x_i \in T$ to its corresponding class label $l_j \in L$.
- $c : L \mapsto 2^T$ is a function that maps a given class label $l_j \in L$ to a given set C, such that $C \subseteq T$, which represents the set of instances in T whose class is l_j. Notice that $T = \bigcup_{l \in L} c(l)$. In this notation, 2^T represents the power set of T, that is, the set of all subsets of T , including the empty set and T itself.
- $d : T \times T \mapsto R$ is a distance function (or dissimilarity function), which maps two instances to a real number that represents the distance (or dissimilarity) between them.
- $S = x_1, x_2, ..., x_q$ is a set of q instances, such that $S \subseteq T$. It represents the reduced set of instances resulting from the instance selection process.
- $B = b_1, b_2, ..., b_s$ is a set of s size buckets, such that each bucket b_i contains a set of similar instances $x \in P$.
- AS, is the agreement set of elements from $S - \{x\}$ of a instance x from S, such that each element of AS have the same class label as x and they belong to partial neighbors of $x, AS \subseteq pkn(x, k)$.

IV. The LSH-LDIS Algorithm

Definition. *The density of a given instance $x \in P$, where $P \subseteq T$, is a measure of the spatial concentration of other instances within P around the instance x. Intuitively, it can be viewed as a measure of the average similarity between x and all the instances within P . Thus, the density is formalized by the function $Dens : T \times 2^T \mapsto R$, such as:*

$$Dens(x,P) = \frac{\frac{|P|}{\epsilon + \sum_{y \in P} d(x,y)}}{\epsilon + d(x, centroid(pkn(x,k)))}$$

$$= \frac{|P|}{\epsilon + \sum_{y \in P} d(x,y)} \times \frac{1}{\epsilon + d(x, centroid(pkn(x,k)))}$$

$$\approx \frac{|P|}{\sum_{y \in P} d(x,y)} \times \frac{1}{d(x, centroid(pkn(x,k)))}$$

$$\approx \frac{1}{d(x, centroid(pkn(x,k))) \times \sum_{y \in P} d(x,y)}$$

where d is a given distance function and ϵ is a very small positive constant, 10^{-17}. We omitted this constant in 3rd line and cardinally of set P in 4th line since it is constant normalization term for a fixed k-neighbors . Notice that $Dens(x,P)$ provides the density of the instance x relatively to the set P of instances. In this way, when P is a subset of the whole dataset, $Dens(x,P)$ represents the local density of x, considering the set P. This notion of density was adapted from [19, 20, 21].

In this way, when P is a subset of the whole dataset, $Dens(x,P)$ represents the local density of x, considering the set P. This notion of density was adapted from [19, 20, 21].

Definition. *The LSH(Local Sensitive Hashing) is a function that maps a given instance $x \in T$ to a bucket $b_i \in B$ such that similar instances were assigned in the same bucket and dissimilar instances were assigned to separate buckets, $\mathbf{LSH} : \mathbf{x} \in \mathbf{T} \mapsto \mathbf{b_i} \in \mathbf{B}$ using some similarity function, f. For mapping a instance to a bucket, binary random projections and dot product as similarity function is used. Lets say, we have fixed k bit bucket then a random normal (Gaussian) matrix is defined by $\mathbf{M} \in \mathbf{R^{k \times m}} \in \mathbf{N(0,1)}$. For a instance $\mathbf{x} \in \mathbf{R^{1 \times m}}$ in T, projection $\mathbf{v} \in \mathbf{R^{k \times 1}}$ is the matrix product of $\mathbf{M}$ and $\mathbf{x}$, $\mathbf{Mx^t}$ where $\mathbf{t}$ denotes transpose of x. Then bucket key for instance $\mathbf{x}$, which identify a bucket uniquely is the binary bit string $\mathbf{Key}$ of length k generated from $\mathbf{v}$ vector of k-elements conditioned as*

$$\mathbf{Key_{1 \leq i \leq k}} = \begin{cases} 1, & \text{if } v[i] \geq 0.0 \\ 0, & \text{otherwise} \end{cases}$$

Notes, same random matrix $\mathbf{M}$ is used for all instances until we reinitialize local sensitive hashing.

The LSH-IS algorithm also adopts the notion of partial k-neighborhood from [12].

Definition. *The partial k-neighborhood is formalized by the function PKN: $T \times N_1 \mapsto 2^T$ that maps a given instance $x \in T$ and a given $k \in N_1$ $(k \geq 1)$ to a given set C, such that $C \subseteq (c(l(x)) - \{x\})$, which represents the set of the k nearest neighbors of x, in $c(l(x))$ (excepting x_i). Since the resulting set C includes only the neighbors that have a given class label, it defines a partial k-neighborhood.*

Finally, the LSH-IS algorithm also adopts the notion of local density ordering (LDO).

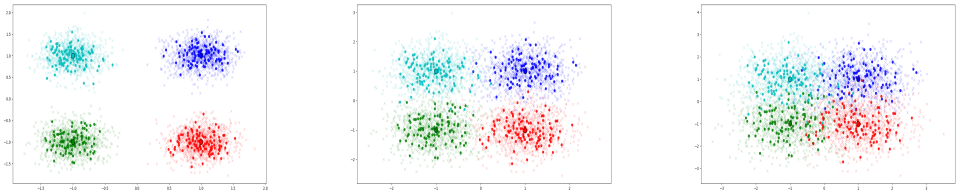
Definition. *The local density ordering is formalized by the function $LDO : T \mapsto N$ that maps a given instance $x \in T$ to a value $o \in N$ that represents the ordering of the instance x when the set $c(l(x))$ is sorted according to the local density of the instances within the set $c(l(x))$. Thus, the LDO of a given instance x represents how dense x is, in comparison with the other instances within $c(l(x))$*

Notice that when all instances within a set $c(l(x))$ have different local densities, the local density ordering of an instance x represents the number of instances within $c(l(x))$ whose local density is less than $Dens(x, c(l(x)))$. However, when the instances in a set $G \subseteq c(l(x))$ have the same local density, the function LDO assigns a different local density ordering for each of them. That is, supposing that $G = y0, y1, ..., ys$, the LDO of the instances in G would be: $LDO(y0) = ld_G, LDO(y1) = ld_G + 1, ..., LDO(ys) = ld_G + s$; where $ld_G = |i | i \in c(l(x)) \wedge Dens(i, c(l(x))) < Dens(y0, c(l(x)))|$. Thus, the LDO is able to distinguish instances with the same local density. The XLDIS algorithm assumes that the local density ordering, LDO(x), of a given instance x is proportional to its capability of representing information about its neighborhood. Considering this, for each $x \in c(l)$, the algorithm defines $toAvoid_x$ as false, indicating that this instance should be considered as a candidate for selection. Then, the algorithm sorts the c(l) set, in a descending order, according to the LDO of the instances. In the next step, for each $l \in L$, the LSH-LDIS algorithm verifies if x should be considered as a candidate for selection (if $Avoid_x = true$). If this is the case, it verifies if there is some instance $y \in pkn(x,k)$ (where k is arbitrarily chosen by the user), such that $LDO(y) > LDO(x)$. If this is not the case, this means that x has the highest local density ordering in its partial k-neighborhood and, due to this, x should be included in S. Besides that, the algorithm considers that every instance $y \in pkn(x,k)$, such that

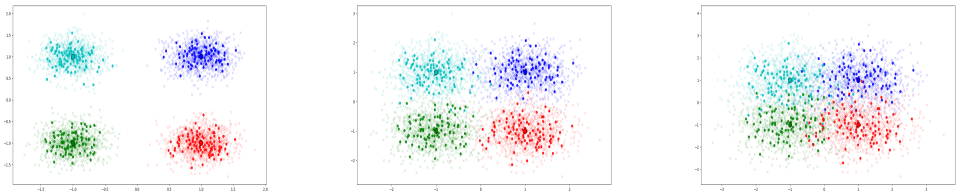
$LDO(y) < LDO(x)$ should be avoided in further analysis, because they have a neighbor (x) with a higher LDO that was already included in the final set. The Algorithm 1 formalizes this strategy.

The most expensive steps of the algorithm involve determining the local density and the partial k-neighborhood of each instance of a given set $c(l)$. Determining the partial density of every instance of a given set $c(l)$ is a process whose time complexity is proportional to $O(|c(l)|^2)$. The time complexity of determining the partial k-neighborhood is equivalent. An efficient implementation of the Algorithm 1 could calculate the partial k-neighborhood and the partial density of each instance of a given set $c(l)$ just once, in a first step within the main loop, and use this information for further calculations. Considering this, the time complexity of instance selection would be proportional to $O(\sum_{l \in L} |c(l)|^2)$. Similarly, time complexity of outlier or noise removal step would be proportional to $O(|S|^2)$. However, run time would be dominated by selection step. Thus, the LSH-IS algorithm has a time complexity equivalent to the time complexity of the XLIDS algorithm.

V. EXPERIMENTS

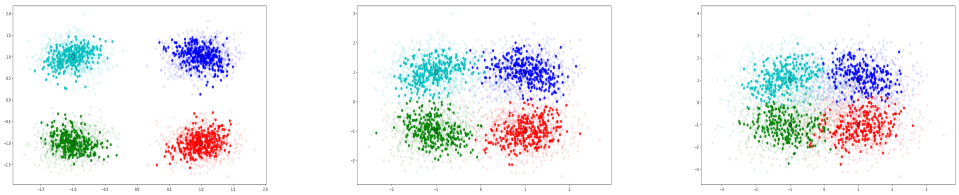


(a) LIDS, std=0.25 (b) LIDS, std=0.50 (c) LIDS, std=0.75

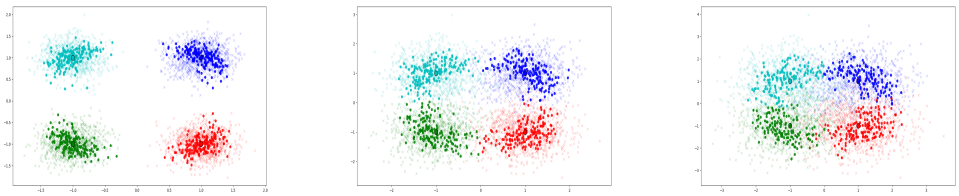


(d) XLIDS, std=0.25 (e) XLIDS, std=0.50 (f) XLIDS, std=0.75

Fig. 1: Limitation of LIDS and XLIDS in removing out-liar and noisy data points. LIDS, and XLIDS algorithm applied on 4096 instances of 2-dimensional data points having 4 cluster centered at [1,1], [1,-1], [-1, 1], [-1,-1] with varying standard deviation(std). Selected points are marked as filled round ball, removed points are marked as faded cross ($\times$), and cluster centers are focused as big round ball.



(a) LSH-IS, k=3, std=0.25 (b) LSH-IS, k=3, std=0.50 (c) LSH-IS, k=3, std=0.75



(d) LSH-IS, k=5, std=0.25 (e) LSH-IS, k=5, std=0.50 (f) LSH-IS, k=5, std=0.75

Fig. 2: Effectiveness of LSH-IS in removing out-liar and noisy data points. LSH-IS algorithm applied on 4096 instances of 2-dimensional data points having 4 cluster centered at [1,1], [1,-1], [-1, 1], [-1,-1] with varying standard deviation(std). Selected points are marked as filled round ball, removed points are marked as faded cross ($\times$), and cluster centers are focused as big round ball. k is the number of nearest neighbors to consider in partial-k-neighbors computations.

TABLE I: Details of the datasets used in the evaluation process.

Dataset	Instances	Attributes	Classes
Breast Cancer	286	10	2
Cars	1728	6	4
E. Coli	336	8	8
Iris	150	5	3
Letter	20000	17	26
Lung Cancer	32	57	3
Mushroom	8124	23	2
Parkinson	195	23	2
Soybean	683	36	19
Voting	435	17	2
Wine	178	14	3
Zoo	101	18	7

In the first experiment, we will show visually when LIDS and its extended version XLIDS fails to detect outliar and noisy data points, hence removal of those data points. Syntactic 2d points with four cluster of varying standard deviation were used for this purpose. The more standard deviation the more overlapping between these inter cluster data points and hence noisy data points. From fig. 1, we can clearly see that LIDS and XLIDS fails to recognize and hence removal of noisy data points if dataset contains noisy overlapped instances of other clusters. To address this issues, we propose LSH-IS algorithm of instance selection that not only removes those noisy data points but also retains quick computation advantages of XLIDS without too much degradation of performance. We can evidently see the clear separation between clusters among the selected points in fig. 2 even in the presence of noisy data points, because our algorithm incorporate noisy points detection mechanism and hence removal of those noisy points.

For evaluating our approach, we compared the XLDIS algorithm in a classification task, with 6 important instance selection algorithms provided by the literature: DROP3, ENN, ICF, LSBo, LSSm and the original LDIS. We considered 12 well-known datasets: breast cancer, cars, E. Coli, iris, letter, lung cancer, mushroom, parkinson, soybean, congressional voting records, wine and zoo. All datasets were obtained from the UCI Machine Learning Repository[1]. In Table I, we present the details of the datasets that were used.

We use three standard measures to evaluate the performance of IS algorithms: accuracy, reduction and effectiveness proposed in [12]. Following [6, 12], we assume:

$$Accuracy = \frac{Success(Test)}{|Test|} \tag{1}$$

and

$$Reduction = \frac{|T| - |S|}{|T|} \tag{2}$$

where Test is a given set of instances that are selected for being tested in a classification task, and Success(Test) is the number of instances in Test correctly classified in the classification task. Besides that, we consider effectiveness as a measure of the degree to which an instance selection algorithm is successful in producing a small set of instances that allows a high classification accuracy of new instances. Thus, we consider $effectiveness = accuracy \times reduction$, as per [12].

To evaluate the classification accuracy of new instances in each respective dataset, we adopted the k-Nearest Neighbors (KNN) algorithm [13], considering k = 3, as assumed in [6, 12]. Besides that, following [12], the accuracy and reduction were evaluated in an n-fold cross-validation scheme, where $n = 10$. Thus, firstly a dataset is randomly partitioned in 10 equally sized subsamples. From these subsamples, a single subsample is selected as validation data (Test), and the union of the remaining 9 subsamples is considered the initial training set(ITS). Next, an instance selection algorithm is applied for reducing the ITS, producing the reduced training set(RTS). At this point, we can measure the reduction of the dataset. Finally, the RTS is used as the training set for the KNN algorithm, to classify the instances in Test. At this point, we can measure the accuracy achieved by the KNN, using RTS as the training set. This process is repeated 10 times, with each subsample used once as Test. The 10 values of accuracy and reduction are averaged to produce, respectively, the average accuracy (AA) and average reduction (AR). The average effectiveness is calculated by considering AA and AR. We did not perform experiment for other algorithms, directly adopted experimental result from XLIDS [21] paper, Since we are working on the same dataset following exactly the same approach mentioned in XLIDS paper. Tables II, III and IV report, respectively, the resulting AA, AR, and AE of each combination of dataset and instance selection algorithm; the best results for each dataset being marked in bold typeface.

In this experiment, following [12], we adopted $k = 3$ for our algorithm LSH-IS which also adopted for

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

DROP3, ENN, ICF, and LDIS. Besides that, we adopted the following distance function $d : T \times T \mapsto R$:

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

with

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

where

$$\theta_j(x, y) = \begin{cases} 1, & \text{if } x_j = y_j \\ 0, & \text{if } x_j \neq y_j \end{cases}$$

TABLE II: Comparison of the accuracy achieved by the training set produced by each algorithm, for each dataset.

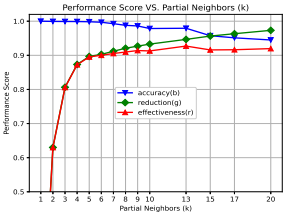
Dataset	LSBo	DROP3	ICF	ENN	LSSm	LDIS	XLDIS	LSH-IS
Breast Cancer	0.61	0.73	0.72	**0.74**	**0.74**	0.69	0.69	0.73
Cars	0.65	0.75	**0.76**	**0.76**	**0.76**	**0.76**	**0.76**	0.71
E. Coli	0.80	0.83	0.81	**0.86**	0.85	0.85	0.85	0.76
Iris	0.94	**0.97**	0.94	**0.97**	0.96	0.96	0.95	0.85
Letter	0.75	0.87	0.80	**0.92**	**0.92**	0.77	0.75	0.79
Lung Cancer	0.32	0.31	0.41	0.34	0.45	0.38	0.43	**0.52**
Mushroom	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
Parkinsons	**0.85**	0.83	0.81	**0.85**	**0.85**	0.80	0.81	0.78
Soybean	0.63	0.85	0.84	0.90	**0.91**	0.78	0.73	0.77
Voting	0.89	**0.92**	0.91	**0.92**	**0.92**	0.91	0.91	0.89
Wine	0.78	0.70	0.73	0.74	0.76	0.69	0.71	**0.91**
Zoo	0.71	**0.91**	0.88	0.88	0.90	0.82	0.66	0.81

TABLE III: Comparison of the reduction achieved by the training set produced by each algorithm, for each dataset.

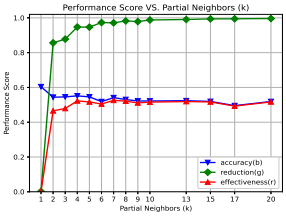
Dataset	LSBo	DROP3	ICF	ENN	LSSm	LDIS	XLDIS	LSH-IS
Breast Cancer	0.73	0.77	0.85	0.29	0.14	0.88	**0.90**	0.89
Cars	0.74	0.88	0.82	0.18	0.12	0.86	0.89	**0.90**
E. Coli	0.82	0.71	0.86	0.16	0.09	0.90	0.92	**0.92**
Iris	**0.93**	0.72	0.58	0.04	0.06	0.89	0.91	0.90
Letter	0.83	0.68	0.80	0.05	0.04	0.83	0.88	**0.88**
Lung Cancer	0.52	0.70	0.74	0.59	0.17	0.85	0.86	**0.87**
Mushroom	**0.99**	0.86	0.94	0.0	0.0	0.86	0.90	0.90
Parkinsons	0.87	0.71	0.75	0.15	0.12	0.81	0.87	**0.90**
Soybean	0.83	0.68	0.58	0.09	0.05	0.78	**0.86**	0.78
Voting	0.89	0.79	**0.92**	0.07	0.04	0.77	0.90	0.82
Wine	0.78	0.71	0.78	0.23	0.10	0.87	0.88	**0.90**
Zoo	**0.88**	0.65	0.33	0.06	0.06	0.65	0.82	0.69

TABLE IV: Comparison of the effectiveness achieved by each algorithm, for each dataset.

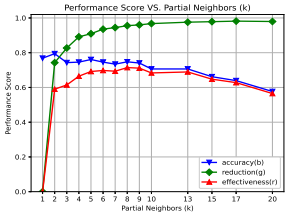
Dataset	LSBo	DROP3	ICF	ENN	LSSm	LDIS	XLDIS	LSH-IS
Breast Cancer	0.45	0.56	0.61	0.21	0.10	0.61	0.62	**0.65**
Cars	0.49	0.66	0.62	0.14	0.09	0.65	**0.68**	0.64
E. Coli	0.66	0.59	0.70	0.14	0.08	**0.77**	**0.77**	0.70
Iris	**0.87**	0.69	0.55	0.04	0.06	0.86	**0.87**	0.76
Letter	0.62	0.59	0.65	0.05	0.03	0.64	0.66	**0.70**
Lung Cancer	0.17	0.22	0.30	0.20	0.08	0.32	0.37	**0.45**
Mushroom	**0.99**	0.86	0.94	0.00	0.00	0.86	0.90	0.90
Parkinsons	**0.74**	0.59	0.61	0.13	0.10	0.65	0.70	0.70
Soybean	0.52	0.58	0.49	0.08	0.04	0.61	**0.64**	0.60
Voting	0.79	0.72	**0.84**	0.07	0.04	0.71	0.82	0.73
Wine	0.61	0.50	0.57	0.17	0.07	0.60	0.62	**0.82**
Zoo	**0.62**	0.59	0.29	0.05	0.05	0.53	0.54	0.56



(a) Mushroom Dataset



(b) Splice Dataset



(c) Cars Dataset

Fig. 3: Impact of partial neighbors(k) on Performances. In each dataset, K-fold where K=5 is used to measure performance scores.

Table II shows that ENN and LSSm achieves the 1st and 2nd highest accuracy respectively in most of the datasets. However, since both ENN and LSSm were designed for removing noisy instances, it does not provide high reduction rates. Besides that, it is important to notice that, for most of the datasets, the difference between the accuracy of LSH-IS and the accuracy achieved by the other algorithms is not significantly large. In cases where the achieved accuracy is lower than the accuracy provided by other algorithms, this can be compensated by a higher reduction. The Table II also shows that, regarding the accuracy, the results achieved by LSH-IS are very similar to those of XLDIS, LDIS. On the other hand, in the wine dataset, LSH-IS provided an accuracy that is significantly higher than that of both XLDIS and LDIS. Table III shows that LSH-IS achieves the highest reduction in most of the datasets, and achieves also the highest average reduction rate. This table also shows that, in some datasets (such as Cars, Letter, Parkinsons), the LSH-IS algorithm achieved a reduction rate that is significantly higher than that of LDIS and slightly higher or equal than that of XLDIS. Finally, Table IV shows that LSH-IS has the highest effectiveness in most of the datasets, as well as the highest average effectiveness. Thus, these results demonstrate that although LSH-IS does not provide always the highest accuracies, it provides the highest reduction rates and the best trade-off between both measures (represented by the effectiveness).

We also carried out experiments for evaluating the impact of the parameter k in the performance of LSH-IS. The Fig.3 represents the average accuracy, average reduction and average effectiveness achieved by LSH-IS as a function of the parameter k in each of the three dataset, with k assuming the values between 1 and 20. These results show that the variation of k has a significant impact on the performance of the algorithm.

The results suggest that, in general, as the value of k increases, the accuracy decreases, the reduction increases, and the effectiveness increases up to a point from which it begins to decrease. This plot also shows that, with k = 13, the LSH-IS algorithm achieves balanced average performance score for both Mushroom and Splice dataset, while for Cars dataset k=8 is the best.

We also carried out a comparison of the running times of the instance selection algorithms considered in our experiments. In this comparison, we applied top two instance selection algorithms from XLIDS paper and our LSH-IS to reduce syntactic datasets consisting of 2d points having four class or clusters. These are D1, D2, D3, and D4 having 10k, 20k, 25k, 35k instances respectively. For conducting the experiments, we used an Intel CoreTM i5-8250U laptop with a 1.60 GHz CPU and 8 GB of RAM. The Fig. 4 shows that, considering these four datasets, the LSH-IS algorithm has almost twice the running time of XLIDS and almost equal running time of LIDS. The Fig. also shows that we can reap benefit of using Local Sensitive Hashing technique if dataset size grows large, then running time of LSH-IS would be lower than LIDS. This result is a consequence of the fact that LDIS deals with the set of instances of each class of the dataset separately, and find k-partial neighbors from these instances which is costly operations. LSH-IS leverages this fact by introducing Local Sensitive Hashing with random binary projection. Another advantages as mentioned earlier is that LSH-IS have notion of outliers and detection mechanism. For this extra advantage, LSH-IS algorithm is worse in run time than XLIDS. We can, of course sacrifice this not so much run time if we want to detect outliers instances. Outlier detection will certainly increase accuracy of underlying model and reduction is guaranteed not worse than XLIDS. Therefore, this algorithm can be considered a trade-off between accuracy and reduction.

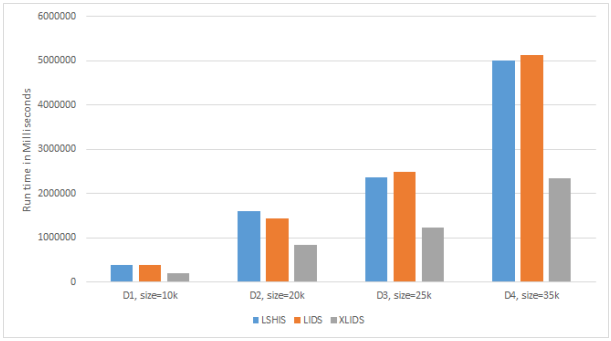


Fig. 4: Execution(Run) time of best three algorithms with syntactic datasets consisting of 2d points having four class or clusters and partial neighbors(k=5) is used.

VI. Conclusion

In this paper, we proposed a new outlier aware efficient algorithm for instance selection, called LSH-IS based on Local Density augmented with faster partial neighbors search technique using local sensitive hashing. It identifies the most representative instances of each class adopting a local search procedure. Due to this, its time complexity is lower than the time complexity of

approaches (ENN, LSBo, LSSm, DROP3, ICF) that identifies boundary instances. Besides that, LSH-IS assumes that the most representative instance in a given neighborhood has the higher local density ordering (LDO) within that neighborhood. The LDO of a given instance x is basically the number of instances whose local density is lower than the local density of x. Our experiments show that LSH-IS provides the best reduction rates and the best balance between accuracy and reduction, with comparatively lower time complexity, compared with other algorithms available in the literature. The experiments also showed that the LSH-IS algorithm is much more aggressive in removing redundant instances than LDIS plus it offers supervised outlier detection and removal mechanism that XLIDS lacks. These features make LSH-IS a promising algorithm for dealing with Big Data. To deal with Big Data with limited memory (RAM) resource, we proposed a batch instance selection based on LSH-IS which may take advantage of multiple computing resources(CPUs). In future works, we plan to investigate strategies for automatically estimating the best value of the parameter k for each problem. Besides that, the performance of LSH-IS encourages the investigation of novel instance selection strategies that are based on other local properties of the dataset. Regarding this point, we plan to investigate how to use centrality measures, typically used in Network Science, for identifying the instances that should be selected.

REFERENCES

[1] Wei Fan and Albert Bifet. Mining big data: current status, and forecast to the future. *ACM SIGKDD explorations newsletter*, 14(2):1–5, 2013.

[2] Salvador García, Julián Luengo, and Francisco Herrera. *Data preprocessing in data mining*. Springer, 2015.

[3] Chien-Hsing Chou, Bo-Han Kuo, and Fu Chang. The generalized condensed nearest neighbor rule as a data reduction method. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 2, pages 556–559. IEEE, 2006.

[4] Huan Liu and Hiroshi Motoda. On issues of instance selection. *Data Mining and Knowledge Discovery*, 6(2):115, 2002.

[5] Henry Brighton and Chris Mellish. Advances in instance selection for instance-based learning algorithms. *Data Min. Knowl. Discov.*, 6(2):153–172, April 2002.

[6] Enrique Leyva, Antonio González, and Raúl Pérez. Three new instance selection methods based on local sets: A comparative study with several approaches from a bi-objective perspective. *Pattern Recognition*, 48(4):1523–1537, 2015.

[7] Wei-Chao Lin, Chih-Fong Tsai, Shih-Wen Ke, Chia-Wen Hung, and William Eberle. Learning to detect representative data for large scale instance selection. *Journal of Systems and Software*, 106:1–8, 2015.

[8] Konstantinos Nikolaidis, John Yannis Goulermas, and QH Wu. A class boundary preserving algorithm for data condensation. *Pattern Recognition*, 44(3):704–715, 2011.

[9] D Randall Wilson and Tony R Martinez. Reduction techniques for instance-based learning algorithms. *Machine learning*, 38(3):257–286, 2000.

[10] Dennis L Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3):408–421, 1972.

[11] Joel Luís Carbonera. An efficient approach for instance selection. In *International conference on big data analytics and knowledge discovery*, pages 228–243. Springer, 2017.

[12] Joel Luis Carbonera and Mara Abel. A density-based approach for instance selection. In *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 768–774. IEEE, 2015.

[13] Thomas Cover and Peter . Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[14] Peter Hart. The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 14(3):515–516, 1968.

[15] Geoffrey Gates. The reduced nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 18(3):431–433, 1972.

[16] I. Tomek. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(6):448–452, 1976.

[17] Yoel Caises, Antonio González, Enrique Leyva, and Raúl Pérez. Combining instance selection methods based on data characterization: An approach to increase their effectiveness. *Inf. Sci.*, 181(20):4780–4798, October 2011.

[18] Javad Hamidzadeh, Reza Monsefi, and Hadi Sadoghi Yazdi. Irahc: instance reduction algorithm using hyperrectangle clustering. *Pattern Recognition*, 48(5):1878–1889, 2015.

[19] Joel Luis Carbonera and Mara Abel. A cognitively

inspired approach for knowledge representation and reasoning in knowledge-based systems. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 4349–4350, 2015.

[20] Joel Luis Carbonera and Mara Abel. Categorical data clustering: a correlation-based approach for unsupervised attribute weighting. In *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*, pages 259–263. IEEE, 2014.

[21] Joel Luis Carbonera and Mara Abel. Extended ontologies: a cognitively inspired approach. In *ONTOBRAS*, 2015.

APPENDIX

Algorithm 1: LSH-IS instance selection algorithm

input : A set instances T, the number k partial neighbors to search, and a threshold(th).
output: A set of instances, S such that $S \subseteq T$

$S \leftarrow \emptyset$;
foreach $l \in L$ **do**
  $Avoid_x \leftarrow false$, for every $x \in c(l)$;
  Reinitialize LSH, implemented with random projections.
  foreach $x \in c(l)$ **do**
    Index instance x, using LSH.
  end
  Sort $c(l)$ in a descending order, according to the LDO of its instances;
  foreach $x \in c(l)$ **do**
    if $Avoid_x = false$ **then**
      foundHigher $\leftarrow$ false;
      foreach $y \in pkn(x, k)$ **do**
        if *LDO(x) < LDO(y)* **then**
          foundHigher $\leftarrow$ true;
        end
      end
      if $\neg$ *foundHigher* **then**
        $S \leftarrow S \cup \{x\}$;
        foreach $y \in pkn(x, k)$ **do**
          if *LDO(x) > LDO(y)* **then**
            $Avoid_y \leftarrow$ true;
          end
        end
      end
    end
  end
end
Reinitialize LSH;
foreach $x \in S$ **do**
  Index instance x using LSH.;
end
$Avoid_x \leftarrow false$, for every $x \in S$;
foreach $x \in S$ **do**
  if $Avoid_x = false$ **then**
    $AS \leftarrow \{y | y \in pkn(x, k) \wedge l(y) = l(x)\}$;
    $fraction \leftarrow \frac{|AS|}{|pkn(x,k)|}$
    if $fraction \leq threshold(th)$ **then**
      $S \leftarrow S - \{x\}$
    end
    if $fraction > threshold(th) \wedge fraction \leq 1.0$ **then**
      $Avoid_y \leftarrow true$, for every $y \in AS$;
    end
  end
end

Algorithm 2: A Batch LSH-IS instance selection algorithm

input : A file path where all instances (T) and their labels located, the number k partial neighbors to search, and a threshold(th).
output: A set of instances, S such that $S \subseteq T$

$S \leftarrow \emptyset$;
$BS \leftarrow$ Size of the batch;
$BN \leftarrow 1 + \lfloor \frac{|T|}{BS} \rfloor$, number of batch to cover all instances in T at least one in the selection process;
for *Round* $\leftarrow 1$ **to** BN **do**
  $B \leftarrow$ Randomly select BS number of instances without replacement from files;
  $CS \leftarrow$ Run Algorithm **??** with B, k, and $threshold(th)$ as parameters.;
  $S \leftarrow S \cup CS$;
  Save intermediate step in files until all instances are processed, if total data instances $|T|$ is too large to handle at once. i.e saving Selected set S and indices of Processed instances both currently selected(CS) and not selected($B - \{CS\}$), then in previous random selection step choose only not processed instances.;
end