Outlier Detection with One-Class Classifiers from ML and KDD

Jeroen H.M. Janssens, Ildiko Flesch, and Eric O. Postma
Tilburg centre for Creative Computing
Tilburg University
Tilburg, The Netherlands
Email: jeroen@jeroenjanssens.com, ildiko.flesch@gmail.com, eric.postma@gmail.com

***Abstract*—The problem of outlier detection is well studied in the fields of Machine Learning (ML) and Knowledge Discovery in Databases (KDD). Both fields have their own methods and evaluation procedures. In ML, Support Vector Machines and Parzen Windows are well-known methods that can be used for outlier detection. In KDD, the heuristic local-density estimation methods LOF and LOCI are generally considered to be superior outlier-detection methods. Hitherto, the performances of these ML and KDD methods have not been compared. This paper formalizes LOF and LOCI in the ML framework of one-class classification and performs a comparative evaluation of the ML and KDD outlier-detection methods on real-world datasets. Experimental results show that LOF and SVDD are the two best-performing methods. It is concluded that both fields offer outlier-detection methods that are competitive in performance and that bridging the gap between both fields may facilitate the development of outlier-detection methods.**

***Keywords*-one-class classification; outlier detection; local density estimation**

I. Introduction

There is a growing interest in the automatic detection of abnormal or suspicious patterns in large data volumes to detect terrorist activity, illegal financial transactions, or potentially dangerous situations in industrial processes. The interest is reflected in the development and evaluation of outlier-detection methods [1], [2], [3], [4]. In recent years, outlier-detection methods have been proposed in two related fields: Knowledge Discovery (in Databases) (KDD) and Machine Learning (ML). Although both fields have considerable overlap in their objectives and subject of study, there appears to be some separation in the study of outlier-detection methods. In the KDD field, the Local Outlier Factor (LOF) method [3] and the Local Correlation Integral (LOCI) method [4] are the two main methods for outlier detection. Like most methods from KDD, LOF and LOCI are targeted to process large volumes of data [5]. In the ML field, outlier detection is generally based on data description methods inspired by k-Nearest Neighbors (kNNDD), Parzen Windows (PWDD), and Support Vector Machines (SVDD), where DD stands for data description [1], [2]. These methods originate from statistics and pattern recognition, and have a solid theoretical foundation [6], [7].

Interestingly, within both fields the evaluation of outlier-detection methods occurs quite isolated from the other field. In the KDD field, LOF and LOCI are rarely compared to ML methods such as kNNDD, PWDD, and SVDD [3], [4] and in the ML field, LOF and LOCI are seldom mentioned. As a case in point, in Hodge and Austin's review of outlier detection methods [8], LOF and LOCI are not mentioned at all, while in a recent anomaly-detection survey [9], these methods are compared on a conceptual level, only. The aim of this paper is to treat outlier-detection methods from both fields on an equal footing by framing them in a common methodological framework and by performing a comparative evaluation. To the best of our knowledge, this is the first time that outlier-detection methods from the fields of KDD and ML are evaluated and compared in a statistically valid way.[1]

To this end we adopt the one-class classification framework [1]. The framework allows outlier-detection methods to be evaluated using the well-known performance measure AUC [11], and to be compared using statistically funded comparison test such as the Friedman test [12] and the post-hoc Nemenyi test [13].

The outlier-detection methods of which the performances are compared are: LOF, LOCI from the field of KDD, and kNNDD, PWDD, and SVDD from the field of ML. In this paper, LOF and LOCI are reformulated in terms of the one-class classification framework. The ML methods have been proposed in terms of the one-class classification framework by De Ridder *et al.* [14] and Tax [1].

The remainder of the paper is organized as follows. Section II briefly presents the one-class classification framework. In Sections III and IV we introduce the KDD and ML outlier-detection methods, respectively, and explain how they compute a measure of outlierness. We describe the set-up of our experiments in Section V and their results in Section VI. Section VII discusses the results in terms of three observations. Finally, Section VIII concludes by stating that the fields of KDD and ML have outlier-detection methods that are competitive in performance and deserve treatment on equal footing.

[1]Hido *et al.* recently compared LOF, SVDD, and several other outlier-detection methods [10]. Unfortunately, their study is flawed because no independent test set and proper evaluation procedure were used.

II. One-Class Classification Framework

In the one-class classification framework, outlier detection is formalized in terms of objects and labels as follows. Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^d$ be the object space and let $\mathcal{Y}$ be the corresponding label space. A dataset $\mathcal{D}$ is a sequence of object-label pairs, i.e., $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\} \subseteq \mathcal{X} \times \mathcal{Y}$.

In one-class classification, only example objects from a single class, the target class, are used to train a classifier. This makes a one-class classifier particularly useful for outlier detection [15]. A one-class classifier f classifies a new object $\mathbf{x}_i$ either as belonging to the target class or to the outlier class.

An object is classified as an outlier when it is very 'dissimilar' from the given target objects. To this end, one-class classifiers generally consist of two components: a dissimilarity measure δ and a threshold θ [1]. A new object $\mathbf{x}_i$ is accepted as a target object when the dissimilarity value δ is less than or equal to the threshold θ, otherwise it is rejected as an outlier object:

$$f(\mathbf{x}_i) = \begin{cases} \text{target} & \text{if } \delta(\mathbf{x}_i, \mathcal{D}_{\text{train}}) \leq \theta, \\ \text{outlier} & \text{if } \delta(\mathbf{x}_i, \mathcal{D}_{\text{train}}) > \theta, \end{cases} \tag{1}$$

where $\mathcal{D}_{\text{train}}$, the training set, is a subset of dataset $\mathcal{D}$.

Each method that is presented in this paper has a different way to compute the dissimilarity measure, which, together with the dataset at hand, determines the optimal threshold. In our experiments, the methods are evaluated on a complete range of thresholds using the AUC performance measure [11].

III. KDD Outlier-Detection Methods

In this section we describe two popular outlier-detection methods from the field of KDD, namely the Local Outlier Factor method (LOF) [3] and the Local Correlation Integral method (LOCI) [4]. Both methods are based on *local* densities, meaning that they consider an object to be an outlier when its surrounding space contains *relatively* few objects (i.e., when the data density in that part of the data space is relatively low).

We frame the KDD outlier detection methods LOF and LOCI into the one-class classification framework [16] by letting them compute a dissimilarity measure δ by: (i) constructing a neighbourhood around $\mathbf{x}_i$, (ii) estimating the density of the neighborhood, and (iii) comparing this density with the neighborhood densities of the neighboring objects. Subsections III-A and III-B explain how the three steps are implemented in LOF and LOCI, respectively.

A. Local Outlier Factor

The first KDD method we describe is the heuristic Local Outlier Factor method (LOF) [3]. The user needs to specify one parameter, k, which represents the number of neighbors constituting the neighborhood used for assessing the local density.

In order to construct the neighborhood of an object $\mathbf{x}_i$, LOF defines the neighborhood border distance d_{border} of $\mathbf{x}_i$ as the Euclidean distance d from $\mathbf{x}_i$ to its k^{th} nearest neighbor $\text{NN}(\mathbf{x}_i, k)$:

$$d_{\text{border}}(\mathbf{x}_i, k) = d(\mathbf{x}_i, \text{NN}(\mathbf{x}_i, k)) .$$

Then, a neighborhood $\mathcal{N}(\mathbf{x}_i, k)$ is constructed, containing all objects $\mathbf{x}_j$ whose Euclidean distance to $\mathbf{x}_i$ is not greater than the neighborhood border distance d_{border}:

$$\mathcal{N}(\mathbf{x}_i, k) = \{\mathbf{x}_j \in \mathcal{D}_{\text{train}} \setminus \{\mathbf{x}_i\} \mid d(\mathbf{x}_i, \mathbf{x}_j) \leq d_{\text{border}}(\mathbf{x}_i, k)\}.$$

To estimate the density of the constructed neighborhood, the reachability distance is introduced. Intuitively, this distance is defined to ensure that a minimal distance between the two objects $\mathbf{x}_i$ and $\mathbf{x}_j$ is maintained, by "keeping" object $\mathbf{x}_i$ outside the neighborhood of object $\mathbf{x}_j$. The use of the reachability distance causes a smoothing effect whose strength depends on the parameter k. The reachability distance d_{reach} is formally given by:

$$d_{\text{reach}}(\mathbf{x}_i, \mathbf{x}_j, k) = \max\{d_{\text{border}}(\mathbf{x}_j, k), d(\mathbf{x}_j, \mathbf{x}_i)\} .$$

It should be noted that the reachability distance d_{reach} is an asymmetric measure.

The neighborhood density ρ of object $\mathbf{x}_i$ depends on the number of objects in the neighborhood, $|\mathcal{N}(\mathbf{x}_i, k)|$, and on their reachability distances. It is defined as:

$$\rho(\mathbf{x}_i, k) = \frac{|\mathcal{N}(\mathbf{x}_i, k)|}{\sum\limits_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i, k)} d_{\text{reach}}(\mathbf{x}_i, \mathbf{x}_j, k)} .$$

Objects $\mathbf{x}_j$ in the neighborhood that are further away from object $\mathbf{x}_i$, have a smaller impact on the neighborhood density $\rho(\mathbf{x}_i, k)$.

In the third step, the neighborhood density ρ of object $\mathbf{x}_i$ is compared with those of its surrounding neighborhoods. The comparison results in a dissimilarity measure δ_{LOF} and requires the neighborhood densities $\rho(\mathbf{x}_j, k)$ of the objects $\mathbf{x}_j$ that are inside the neighborhood of $\mathbf{x}_i$. The dissimilarity measure δ_{LOF} is defined formally as:

$$\delta_{\text{LOF}}(\mathbf{x}_i, k, \mathcal{D}_{\text{train}}) = \frac{\sum\limits_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i, k)} \frac{\rho(\mathbf{x}_j, k)}{\rho(\mathbf{x}_i, k)}}{|\mathcal{N}(\mathbf{x}_i, k)|}.$$

An object which lies deep inside a cluster gets a local outlier factor dissimilarity value of around 1 because it has a neighborhood density equal to its neighbors. An object which lies outside a cluster has a relatively low neighborhood density and gets a higher local outlier factor.

B. Local Correlation Integral

The Local Correlation Integral method (LOCI) [4] was proposed as an improvement over LOF. More specifically, the authors state that the choice of the neighborhood size, k, in LOF is non-trivial and may lead to erroneous outlier detections. LOCI is claimed to be an improvement over LOF because it considers the local density at multiple scales or levels of granularity. LOCI achieves this by iteratively performing the three steps, each time using a neighborhood of increasing radius $r \in \mathbb{R}^+$. We denote the set of relevant radii as $\mathcal{R}$.

Another difference with LOF is that LOCI defines two neighborhoods for an object $\mathbf{x}_i$: (i) the extended neighborhood, $\mathcal{N}_{\text{ext}}$, and (ii) the local neighborhood, $\mathcal{N}_{\text{loc}}$. The extended neighborhood of an object $\mathbf{x}_i$ contains all objects $\mathbf{x}_j$ that are within radius r from $\mathbf{x}_i$:

$$\mathcal{N}_{\text{ext}}(\mathbf{x}_i, r) = \{\mathbf{x}_j \in \mathcal{D}_{\text{train}} \mid d(\mathbf{x}_j, \mathbf{x}_i) \leq r\} \cup \mathbf{x}_i,$$

and the (smaller) local neighborhood contains all objects that are within radius αr from object $\mathbf{x}_i$:

$$\mathcal{N}_{\text{loc}}(\mathbf{x}_i, r, \alpha) = \{\mathbf{x}_j \in \mathcal{D}_{\text{train}} \mid d(\mathbf{x}_j, \mathbf{x}_i) \leq \alpha r\} \cup \mathbf{x}_i,$$

where α defines the ratio between the two neighborhoods ($\alpha \in (0, 1]$).

In LOCI, the density of the local neighborhood of an object $\mathbf{x}_i$ is denoted by $\rho(\mathbf{x}_i, \alpha r)$, and is defined as $|\mathcal{N}_{\text{loc}}(\mathbf{x}_i, r, \alpha)|$.

The extended neighborhood of an object $\mathbf{x}_i$ has a density $\hat{\rho}(\mathbf{x}_i, r, \alpha)$, which is defined as the average density of the local neighborhoods of all objects in the extended neighborhood of object $\mathbf{x}_i$. In formal terms:

$$\hat{\rho}(\mathbf{x}_i, r, \alpha) = \frac{\sum\limits_{\mathbf{x}_j \in \mathcal{N}_{\text{ext}}(\mathbf{x}_i, r)} \rho(\mathbf{x}_j, \alpha r)}{|\mathcal{N}_{\text{ext}}(\mathbf{x}_i, r)|}.$$

The local neighborhood density of object $\mathbf{x}_i$ is compared to the extended neighborhood density by means of the multi-granularity deviation factor (MDEF):

$$\text{MDEF}(\mathbf{x}_i, r, \alpha) = 1 - \frac{\rho(\mathbf{x}_i, \alpha r)}{\hat{\rho}(\mathbf{x}_i, r, \alpha)}.$$

An object which lies deep inside a cluster has a local neighborhood density equal to its neighbors and therefore gets an MDEF value around 0. The MDEF value approaches 1 as an object lies more outside a cluster.

To determine whether an object is an outlier, LOCI introduces the normalized MDEF:

$$\sigma_{\text{MDEF}}(\mathbf{x}_i, r, \alpha) = \frac{\sigma_{\hat{\rho}}(\mathbf{x}_i, r, \alpha)}{\hat{\rho}(\mathbf{x}_i, r, \alpha)},$$

where $\sigma_{\hat{\rho}}(\mathbf{x}_i, r, \alpha)$ is the standard deviation of all $\rho(\mathbf{x}_j, \alpha r)$ in $\mathcal{N}_{\text{ext}}(\mathbf{x}_i, r)$. The normalized MDEF becomes smaller when the local neighborhoods have the same density. Intuitively, this causes a cluster of uniformly distributed objects to have a tighter decision boundary than, for example, a Gaussian distributed cluster.

We define the dissimilarity measure δ_{LOCI} as the maximum ratio of MDEF to σ_{MDEF} of all radii $r \in \mathcal{R}$:

$$\delta_{\text{LOCI}}(\mathbf{x}_i, \alpha, \mathcal{D}_{\text{train}}) = \max_{r \in \mathcal{R}} \left\{ \frac{\text{MDEF}(\mathbf{x}_i, r, \alpha)}{\sigma_{\text{MDEF}}(\mathbf{x}_i, r, \alpha)} \right\}.$$

IV. ML Outlier-Detection Methods

In this section we briefly discuss the outlier-detection methods from the field of Machine Learning. The methods k-Nearest Neighbor Data Description, Parzen Windows Data Description, and Support Vector Domain Description are explained in Sections IV-A, IV-B, and IV-C, respectively.

A. k-Nearest Neighbor Data Description

The k-Nearest Neighbor Data Description method (KNNDD) [14]. The dissimilarity measure computed by KNNDD is simply the ratio between two distances. The first is the distance between the test object $\mathbf{x}_i$ and its k^{th} nearest neighbor in the training set $\text{NN}(\mathbf{x}_i, k)$. The second is the distance between the k^{th} nearest training object and its k^{th} nearest neighbor. Formally:

$$\delta_{\text{KNNDD}}(\mathbf{x}_i, k, \mathcal{D}_{\text{train}}) = \frac{d(\mathbf{x}_i, \text{NN}(\mathbf{x}_i, k))}{d(\text{NN}(\mathbf{x}_i, k), \text{NN}(\text{NN}(\mathbf{x}_i, k), k))}.$$

The KNNDD method is similar to LOF and LOCI in the sense that it locally samples the density. The main difference with LOF and LOCI is that KNNDD is much simpler.

B. Parzen Windows Data Description

The second ML method is the Parzen Windows Data Description (PWDD), which is based on Parzen Windows [6]. PWDD estimates the probability density function of the target class $\mathbf{x}_i$:

$$\delta_{\text{PWDD}}(\mathbf{x}_i, h, \mathcal{D}_{\text{train}}) = \frac{1}{Nh} \sum_{j=1}^{N} K\left(\frac{\mathbf{x}_i - \mathbf{x}_j}{h}\right),$$

where N is $|\mathcal{D}_{\text{train}}|$, h is a smoothing parameter, and K typically is a Gaussian kernel:

$$K(\mathbf{x}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}.$$

The parameter h is optimised using a leave-one out maximum likelihood estimation [14]. Since the dissimilarity measure δ_{PWDD} is a probability and not a distance, the threshold function (cf. Equation 1) for PWDD becomes:

$$f_{\text{PWDD}}(\mathbf{x}_i) = \begin{cases} \text{target} & \text{if } \delta_{\text{PWDD}}(\mathbf{x}_i, h, \mathcal{D}_{\text{train}}) \geq \theta, \\ \text{outlier} & \text{if } \delta_{\text{PWDD}}(\mathbf{x}_i, h, \mathcal{D}_{\text{train}}) < \theta, \end{cases}$$

such that an object with a too low probability of being a target object is rejected as an outlier. It should be noted that PWDD estimates the density globally, while LOF, LOCI, and KNNDD estimate local densities.

Table I: Summary of the features of the KDD and ML outlier-detection methods used in our experiments.

	KDD		ML		
Feature	LOF	LOCI	KNN	PWDD	SVDD
Estimate local density	✔	✔	✔		
Estimate global density				✔	
Domain based					✔

C. Support Vector Domain Description

The third method is the Support Vector Domain Description (SVDD) [2]. We confine ourselves to a brief description of this kernel-based data-description method. The interested reader is referred to [1], [2] for a full description of the SVDD method.

SVDD is a domain-based outlier-detection method inspired by Support Vector Machines [17], and unlike LOF, LOCI, and PWDD, SVDD does not estimate the data density directly. Instead, it finds an optimal boundary around the target class by fitting a non-linearly transformed hypersphere with minimal volume using the kernel trick, such that it encloses most of the target objects. The optimal boundary is found using quadratic programming, where only distant target objects are allowed to be outside the boundary [17], [7]. The dissimilarity measure δ_{SVDD} is defined as the distance between object $\mathbf{x}_i$ and the target boundary. In our experiments we employ a Gaussian kernel whose width, s, is found as described in [2].

V. Experimental Set-up

This section describes the set-up of our experiments where we evaluate and compare the performances of LOF, LOCI, KNNDD, PWDD, and SVDD. For clarity, we have summarized the features of these methods in Table I. In Section V-A we explain how we prepare the multi-class real-world datasets that we use for our one-class experiments. The evaluation involves the calculation of the weighted AUC, which is described in Section V-B. The statistical Friedman and Nemenyi tests, which we use to compare the methods, are presented in Section V-C.

A. Datasets

In order to evaluate the methods on a wide variety of datasets (i.e., varying in size, dimensionality, class volume overlap), we use 24 real-world multi-class datasets from the UCI Machine Learning Repository[2] [19] as redefined as one-class classification datasets by David Tax (`http://ict.ewi.tudelft.nl/~davidt`): *Arrhythmia*, *Balance-scale*, *Biomed*, *Breast*, *Cancer wpbc*, *Colon*, *Delft Pump*, *Diabetes*, *Ecoli*, *Glass Building*, *Heart*, *Hepatitis*, *Housing*, *Imports*, *Ionosphere*, *Iris*, *Liver*, *Sonar*, *Spectf*, *Survival*, *Vehicle*, *Vowel*, *Waveform*, and *Wine*.

[2]Except the Delft Pump dataset which is taken from Ypma [18].

From each multi-class dataset containing a set of classes C, $|C|$ one-class datasets are constructed by relabelling one class as the target class and the remaining $|C|-1$ classes as the outlier class, for all classes separately.

B. Evaluation

We apply the following procedure in order to evaluate a method on a one-class dataset. An independent test set containing 20% of the dataset is reserved. With the remaining 80% a 5-fold cross-validation procedure is applied to optimise the parameters (i.e., $k = 1, 2, \ldots, 50$ for LOF and KNNDD, $\alpha = 0.1, 0.2, \ldots, 1.0$ for LOCI, h for PWDD, and s for SVDD). Each method is trained with the parameter value yielding the best performance, and its AUC performance is evaluated using the independent test set. This procedure is repeated 5 times.

We report on the performances of the methods on an entire multi-class dataset. To this end, the AUCs of the $|C|$ one-class datasets (cf. Section V-A) are averaged, where each one-class dataset is weighted according to the prevalence, $p(c_i)$, of the target class, c_i:

$$\text{AUC}_{\text{weighted}} = \sum_{\forall c_i \in C} \text{AUC}\left(c_i\right) \times p\left(c_i\right),$$

where $p(c_i)$ is defined as the ratio of the number of target objects to the total number of objects in the dataset. The use of a weighted average prevents one-class datasets containing little target objects from dominating the results [20].

C. Comparison

Following Demšar [21], we adopt the statistical Friedman test and the post-hoc Nemenyi test for the comparison of multiple methods on multiple datasets. The Friedman test [12] is used to investigate whether there is a significant difference between the performances of the methods. The Friedman test first ranks the methods for each dataset, where the best performing method is assigned the rank of 1, the second best the rank of 2, and so forth. Then it checks whether the measured average ranks R_j^2 are significantly different from the mean rank, which is 3 in our case. Iman and Davenport proposed the F_F statistic, which is less conservative than the Friedman statistic [22]:

$$F_F = \frac{\left(N-1\right)\chi_F^2}{N\left(m-1\right)-\chi_F^2},$$

where N is the number of datasets (i.e., 24), m is the number of methods (i.e., 5), and χ_F^2 is the Friedman statistic:

$$\chi_F^2 = \frac{12N}{m\left(m+1\right)}\left(\sum_j R_j^2 - \frac{m\left(m+1\right)^2}{4}\right).$$

The F_F statistic is distributed according to the F-distribution with $m-1$ and $\left(m-1\right)\left(N-1\right)$ degrees of freedom.

Table II: The weighted AUC performance in percentages obtained by the Machine Learning and Knowledge Discovery outlier-detection methods on 24 real-world datasets. The corresponding average rank for each method is reported below.

Dataset	LOF	LOCI	KNN	PWDD	SVDD
Arrhythmia	**62.87**	56.70	56.50	50.00	61.76
Balance-scale	94.66	91.26	93.50	94.18	**95.80**
Biomed	78.61	85.04	**88.29**	72.99	88.20
Breast	96.81	**98.31**	96.68	72.24	97.75
Cancer wpbc	**62.57**	58.55	61.37	56.56	60.59
Colon	73.47	42.08	**75.53**	50.00	70.18
Delft Pump	**94.93**	87.27	93.13	92.97	94.43
Diabetes	**68.24**	64.86	65.03	63.14	67.98
Ecoli	**96.74**	96.10	96.53	93.09	93.39
Glass Building	**81.47**	75.79	78.93	77.39	77.32
Heart	61.82	60.88	60.53	56.86	**62.05**
Hepatitis	**66.53**	62.05	64.72	58.73	60.89
Housing	62.86	63.75	**64.56**	61.66	64.24
Imports	80.49	74.24	80.24	80.09	**82.11**
Ionosphere	74.28	68.51	75.47	71.14	**81.38**
Iris	98.28	98.23	98.49	98.26	**99.20**
Liver	**59.57**	59.10	55.53	53.46	59.15
Sonar	**76.89**	66.71	74.22	74.45	75.77
Spectf	60.20	56.11	52.81	51.57	**78.92**
Survival	67.02	64.18	66.64	62.30	**68.12**
Vehicle	75.68	74.99	79.15	78.81	**81.37**
Vowel	97.89	95.22	99.49	**99.56**	99.28
Waveform	90.26	89.17	89.85	86.89	**90.36**
Wine	88.42	87.68	**89.43**	84.20	86.94
Average rank	2.083	3.917	2.625	4.292	2.083

When there is a significant difference, we proceed with the post-hoc Nemenyi test [13], which checks for each pair of methods whether there is a significant different in performance. The performance of two methods is significantly different when the difference between their average ranks is greater or equal to the critical difference:

$$\text{CD} = q_\alpha \sqrt{\frac{m\,(m+1)}{6N}},$$

where q_α is the Studentised range statistic divided by $\sqrt{2}$. . In our case, CD $=\ 1.245$ for $\alpha = 0.05$.

VI. Results

Table II presents the weighted AUC performances of each method on the 24 real-world datasets.

The average ranks of the methods are shown at the bottom of the table. On these 24 real-world datasets, SVDD and LOF perform best, both with an average rank of 2.083. With an average rank of 2.625, KNNDD performs surprisingly well. LOCI and PWDD perform the worst, with average ranks of 3.917 and 4.292, respectively. Interestingly, SVDD seems to perform well on those datasets where LOF performs worse and vice versa. Apparently, both methods are complementary with respect to the nature of the dataset at hand.

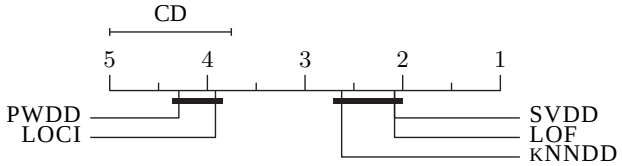

Figure 1: Comparison of all methods against each other with the Nemenyi test. Groups of methods that are not significantly different (at $p = 0.05$) are connected.

To see whether there is a significant difference between these average ranks, we calculate the Friedman statistic, $\chi_F^2 = 48.53$, which results in an F_F statistic of $F_F = 23.52$. With five methods and 24 data sets, F_F is distributed according to the F distribution with $5 - 1 = 4$ and $(5-1) \times (24-1) = 92$ degrees of freedom. The critical value of $F\,(4, 92)$ for $\alpha = 0.05$ is 2.471, so we reject the null-hypothesis, which states that all methods have an equal performance.

We continue with the Nemenyi test, for which the critical distance CD, for $\alpha = 0.05$, is 1.245. We identify two groups of methods. The performances of LOCI and PWDD are significantly worse than that of KNNDD, LOF, and SVDD.

Figure 1 graphically displays the result of the Nemenyi test in a so-called critical difference diagram.

Groups of methods that are not significantly different (at $p = 0.05$) are connected. The diagram reveals that, in terms of performances, the methods examined fall into two clusters. The cluster of best-performing methods consists of SVDD, LOF, and KNNDD. The other cluster contains PWDD and LOCI.

VII. Discussion

We have evaluated five outlier-detection methods from the fields of Machine Learning and Knowledge Discovery in Databases on a real-world of datasets. The performances of the methods have been statistically compared using the Friedman and Nemenyi tests. From the obtained experimental results we make three main observations. We describe each observation separately and provide possible reasons for each of them below.

A. Observation 1: Local density estimates outperform global density estimates

The first observation we make is that PWDD performs significantly worse than LOF. This is to be expected, since PWDD performs a global density estimate. Such an estimation becomes problematic when there exist large differences in the density. Objects in sparse clusters will be erroneously classified as outliers.

LOF and LOCI overcome this problem by performing an additional step. Instead of using the density estimate as a

dissimilarity measure, they locally compare the density with the neighborhood. This produces an estimate which is both relative and local, and enables LOF and LOCI to cope with different densities across different subspaces. For LOF, the local density estimate results in a better performance. For LOCI, however, this is not the case. Possible reasons for this are discussed in the second observation below.

B. *Observation 2:* LOF *outperforms* LOCI

The second observation we make is that LOCI is outperformed by both LOF and KNNDD. This is unexpected because LOCI, just like LOF, considers local densities. Moreover, LOCI performs a multi-scale analysis of the dataset, whereas LOF does not.

We provide two possible reasons for the relative weak performance of LOCI. The first possible reason is that LOF considers three consecutive neighborhoods to compute the dissimilarity measure. LOCI, instead, considers two neighborhoods, only. The three-fold density analysis of LOF is more profound than the two-fold analysis of LOCI and therefore yields a better estimation of the data density.

The second possible reason for the observed results is that LOCI constructs a neighborhood with a given radius, and not with a given number of objects. For small radii, the extended neighborhood may contain one object only, implying that there may be no deviation in the density and that outliers might be missed at a small scale. LOF and KNNDD, on the other hand, do not suffer from this because both methods construct a neighborhood with a given number of objects.

C. *Observation 3: Domain-based and Density-based methods are competitive*

The third observation we make is that domain-based (SVDD) and density-based (LOF) methods are competitive in performance.

To obtain good estimates, density-based methods require large datasets, especially when the object space is of high dimensionality. This implies that in case of sparsely sampled datasets, density-based methods may fail to detect outliers [23]. SVDD describes only the domain in the object space (i.e., it defines a closed boundary around the target class), and does not estimate the complete data density. As a consequence, SVDD is less sensitive to an inaccurate sampling and better able to deal with small sample sizes [1].

VIII. Conclusion

This paper evaluates and compares outlier-detection methods from the fields of Knowledge Discovery in Databases (KDD) and Machine Learning (ML). The KDD methods LOF and LOCI and the ML methods KNNDD, PWDD, and SVDD have been framed into the one-class classification framework, to allow for an evaluation using the AUC performance measure, and a statistical comparison using the Friedman and Nemenyi tests.

In our experimental comparison, we have determined that the best performing methods are KNNDD, LOF, and SVDD. These outlier-detection methods originate from the fields of KDD and ML. Our findings indicate that methods developed in both fields are competitive and deserve treatment on equal footing.

Framing KDD methods in ML-based frameworks such as the one-class classification framework, facilitates the comparison of methods across fields and may lead to novel methods that combine ideas of both fields. For instance, our results suggest that it may be worthwhile to develop outlier-detection methods that combine elements of domain-based and local density-based methods.

We conclude that the fields of KDD and ML offer outlier-detection methods that are competitive in performance and that bridging the gap between both fields may facilitate the development of outlier-detection methods. We identify two directions for future research.

The first direction is to investigate the complementarity of LOF and SVDD with respect to the nature of the dataset. The relative strengths of both methods appear to depend on the characteristics of the dataset. Therefore, investigating which dataset characteristics determine the performances of LOF and SVDD is useful.

The second direction is to combine the best of both fields. For example, to extend LOF with a kernel-based domain description.

Acknowledgment

This work has been carried out as part of the Poseidon project under the responsibility of the Embedded Systems Institute (ESI), Eindhoven, The Netherlands. This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK03021 program. The authors would like to thank the anonymous reviewers and Hans Hiemstra for their critical and constructive comments and suggestions.

References

[1] D. Tax, "One-class classification: Concept-learning in the absence of counter-examples," Ph.D. dissertation, Delft University of Technology, Delft, The Netherlands, June 2001.

[2] D. Tax and R. Duin, "Support vector domain description," *Pattern Recognition Letters*, vol. 20, no. 11-13, pp. 1191–1199, 1999.

[3] M. Breunig, H. Kriegel, R. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Record*, vol. 29, no. 2, pp. 93–104, 2000.

[4] S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos, "LOCI: Fast outlier detection using the local correlation integral," in *Proceedings of the 19th International Conference on Data Engineering*, Bangalore, India, March 2003, pp. 315–326.

[5] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "Knowledge discovery and data mining: Towards a unifying framework," *Knowledge discovery and data mining*, pp. 82–88, 1996.

[6] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics*, pp. 1065–1076, 1962.

[7] B. Scholkopf and A. Smola, *Learning with kernels*. MIT press Cambridge, MA, USA, 2002.

[8] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, October 2004.

[9] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[10] S. Hido, Y. Tsuboi, H. Kashima, M. Sugiyama, and T. Kanamori, "Inlier-based outlier detection via direct density ratio estimation," in *Eighth IEEE International Conference on Data Mining, 2008. ICDM'08*, 2008, pp. 223–232.

[11] A. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[12] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, pp. 675–701, 1937.

[13] P. Nemenyi, "Distribution-free multiple comparisons," Ph.D. dissertation, Princeton, 1963.

[14] D. de Ridder, D. Tax, and R. Duin, "An experimental comparison of one-class classification methods," in *Proceedings of the Fourth Annual Conference of the Advanced School for Computing and Imaging*. Delft, The Netherlands: ASCI, June 1998, pp. 213–218.

[15] N. Japkowicz, "Concept-learning in the absence of counter-examples: An autoassociation-based approach to classification," Ph.D. dissertation, Rutgers University, New Brunswick, NJ, October 1999.

[16] J. Janssens and E. Postma, "One-class classification with LOF and LOCI: An empirical comparison," in *Proceedings of the 18th Annual Belgian-Dutch Conference on Machine Learning*, Tilburg, The Netherlands, May 2009, pp. 56–64.

[17] V. Vapnik, *The nature of statistical learning theory*. Springer-Verlag, NY, USA, 1995.

[18] A. Ypma, "Learning methods for machine vibration analysis and health monitoring," Ph.D. dissertation, Delft University, 2001.

[19] A. Asuncion and D. Newman. (2007) UCI machine learning repository. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[20] K. Hempstalk and E. Frank, "Discriminating against new classes: One-class versus multi-class classification," in *Proc 21st Australasian Joint Conference on Artificial Intelligence*, ser. Auckland, New Zealand. Springer, 2008.

[21] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[22] R. Iman and J. Davenport, "Approximations of the critical region of the Friedman statistic," in *Annual meeting of the American Statistical Association*, vol. 12, 1979.

[23] C. Aggarwal and P. Yu, "Outlier detection for high dimensional data," *ACM SIGMOD Record*, vol. 30, no. 2, pp. 37–46, 2001.