Nearest-Neighbor and Clustering based Anomaly Detection Algorithms for RapidMiner

Mennatallah Amer[1] and Markus Goldstein[2]

[1]Department of Computer Science and Engineering
German University in Cairo, Egypt
[2]German Research Center for Artificial Intelligence (DFKI GmbH)
D-67663 Kaiserslautern, Germany
Markus.Goldstein@dfki.de

Abstract

Unsupervised anomaly detection is the process of finding outlying records in a given dataset without prior need for training. In this paper we introduce an *anomaly detection extension* for RapidMiner in order to assist non-experts with applying eight different nearest-neighbor and clustering based algorithms on their data. A focus on efficient implementation and smart parallelization guarantees its practical applicability. In the context of clustering-based anomaly detection, two new algorithms are introduced: First, a global variant of the cluster-based local outlier factor (CBLOF) is introduced which tries to compensate the shortcomings of the original method. Second, the local density cluster-based outlier factor (LDCOF) is introduced which takes the local variances of clusters into account. The performance of all algorithms have been evaluated on real world datasets from the UCI machine learning repository. The results reveal the strengths and weaknesses of the single algorithms and show that our proposed clustering based algorithms outperform CBLOF significantly.

1 Introduction

Anomaly detection is the task of finding anomalous or outlying records in datasets, whose behavior does not conform with the behavior of the majority. These records have been of increasing interest in many application domains, because their presence could indicate an unauthorized access of a system, credit

card fraud, a failure in a part of a monitored system or a diagnosis of an unknown disease.

There are many attempts to solve the anomaly detection problem. The approaches that are more widely applicable are unsupervised algorithms as they do not need labeled training data meeting the requirements of practical systems. The presented *anomaly detection extension* for RapidMiner[1] implements most of the commonly used unsupervised anomaly detection algorithms. In contrast to supervised machine learning, there is up to now no freely available toolkit such as the presented extension in the anomaly detection domain. Now, non-experts can easily integrate the implemented operators into complex processes using the intuitive graphical user interface of RapidMiner. The algorithms were implemented efficiently in order to ease the comparison of the approaches for researchers as well as making it possible for non-experts, analysts and researchers to freely use the algorithms and perform experiments on larger datasets.

Anomaly detection algorithms could either be *global* or *local*. Global approaches refer to the techniques in which the anomaly score is assigned to each instance with respect to the entire dataset. On the other hand, the anomaly score of local approaches represent the outlierness of the data point with respect to its direct neighborhood. The local approaches can detect outliers that are ignored using global approaches, especially in case of a varying densities within a dataset.

The anomaly detection extension contains two categories of approaches: nearest-neighbor based and clustering based algorithms. Algorithms in the first category assume that outliers lie in sparse neighborhoods and that they are distant from their nearest neighbors [1]. The second category operates on the output of clustering algorithms being thus much faster in general. The extension contains the following algorithms:

- Nearest-neighbor based algorithms
    1. k-NN Global Anomaly Score
    2. Local Outlier Factor (LOF)
    3. Connectivity based Outlier Factor (COF)
    4. Local Outlier Probability (LoOP)
    5. Influenced Outlierness (INFLO)
    6. Local Correlation Integral (LOCI)
- Clustering based algorithms
    1. Cluster based Local Outlier Factor (CBLOF)
    2. Local Density Cluster based Outlier Factor (LDCOF)

A variant of CBLOF, unweighed-CBLOF, is incorporated into the *Cluster based Local Outlier Factor (CBLOF)* operator.

[1] For source code and binaries see `http://madm.dfki.de/rapidminer/anomalydetection`.

The organization of the remainder of the paper is as follows: In Section 2, a quick overview of the implemented nearest-neighbor algorithms will be given. Then in Section 3, the clustering based algorithms are described with a focus on our new approaches. Section 4 describes some optimizations that were implemented including a smart parallelization for the nearest-neighbor based algorithms. In Section 5, the experiments that were performed on real world data to evaluate and compare the algorithms are shown and finally Section 6 concludes.

2 Nearest-Neighbor based Algorithms

Nearest-neighbor based algorithms assign the anomaly score of data instances relative to their neighborhood. They assume that outliers are distant from their neighbors or that their neighborhood is sparse. The first assumption corresponds to k-NN which is a global anomaly detection approach, while the second assumption refers to local density based approaches.

The k-NN global anomaly score is one of the most commonly used nearest-neighbor based algorithms. The anomaly score is either set to the average distance of the k-nearest-neighbors as proposed in [2] or to the distance to the k^{th} neighbor like the algorithm proposed in [3]. Since the first method is much more robust to statistical fluctuations it should be preferred and it is also the method of choice in our experiments later on.

The first local density based approach is the local outlier factor (LOF) [4], which seems to be the most used local method today. It compares the local density of a data instance to that of its neighbors. The local density of a data instance is inversely proportional to the average distance to the k-nearest-neighbors. The LOF score is set to the ratio of the local density of the data instance to the average local density of its neighbors.

This results in normal data having an LOF score of approximately equal to 1, while outliers have scores greater than 1. This is explained by the fact that if the data lies in a dense region, its local density would be similar to that of its neighbors leading to a ratio of 1. For a sufficiently large dataset an LOF score of up to 2 would indicate that the data instance is normal.

Several variants of the LOF algorithm are implemented in the extension. The advantages of LOF include the ease of the interpretation of its score and its ability to capture outliers that were previously unseen by the global approaches. Each of the LOF variants have a different motivation in order to overcome a disadvantage that the authors of the following algorithms observed in the original LOF, while maintaining its advantages. The connectivity based outlier factor (COF) [5] was proposed in order to handle outliers deviating from spherical density patterns, for example straight lines. Influenced outlierness (INFLO) [6] was introduced in order to give more accurate results in case of clusters with varying densities that lie near to each other. This is accomplished

by taking more neighbors into account, namely the reverse k-nearest-neighbors set (RNN_k). $RNN_k(p)$ is the set of objects that has the data instance p in its k-neighborhood set. Local outlier probability (LoOP) [7] incorporates some statistical concepts in order to output the final score as a probability that a particular data instance is a local density outlier. These probabilities facilitate the comparison of a data instance with data in the same dataset as well as in other datasets.

The local correlation integral (LOCI) [8] algorithm is also a variant of LOF. In contrast to LOF, the density of a data instance in LOCI is proportional to the number of objects within a particular radius, the r-neighborhood. For each data instance, the neighborhood is grown from a minimum up to all instances in the dataset. The main motive for this approach is that it doesn't have any crucial parameters like k in the previous approaches and should automatically determine whether a local or global anomaly detection problem has to be solved. This eases the application of the algorithm by the end users. Unfortunately it also causes an increase in both, time $O(n^3)$ and space complexity $O(n^2)$, restricting its use to very small datasets.

3 Clustering based Algorithms

The process of arranging similar objects into groups is referred to as clustering [1]. Clustering based anomaly detection techniques operate on the output of clustering algorithms, e.g. the well-known k-means algorithm. They assume that anomalous instances either lie in sparse and small clusters, far from their cluster centroid or that they are not assigned to any cluster at all. The algorithms that were implemented in our extension use the output of any good clustering algorithm already available in RapidMiner. The initial step followed by these algorithms is to classify the clusters into small and large clusters. The user has the choice to select whether this partitioning is implemented similar to what was proposed in [9] using two parameters α and β or using a single parameter γ similar to the work in [10].

The CBLOF scores, calculated using Equation 1, assign an anomaly score based on the distance to the nearest large cluster multiplied by the size of the cluster the object belongs to. Figure 1 illustrates this concept. Point p lies in the small cluster C_2 and thus the score would be equal to the distance to C_1 which is the nearest large cluster multiplied by 5 which is the size of C_2.

$$CBLOF(p) = \begin{cases} |C_i| \cdot \min(d(p, C_j)) \text{ if } C_i \in SC \text{ where } p \in C_i \text{ and } C_j \in LC \\ |C_i| \cdot d(p, C_i) \text{ if } C_i \in LC \text{ where } p \in C_i \end{cases} \tag{1}$$

The authors of [9] claim that weighting by the size of the cluster makes this method local. However, to our understanding this does not make the method

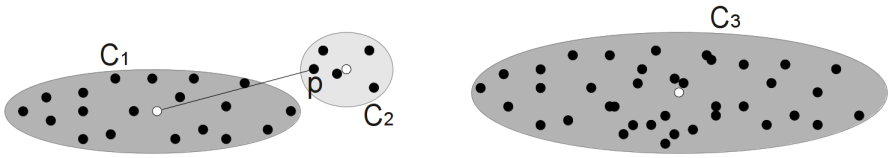
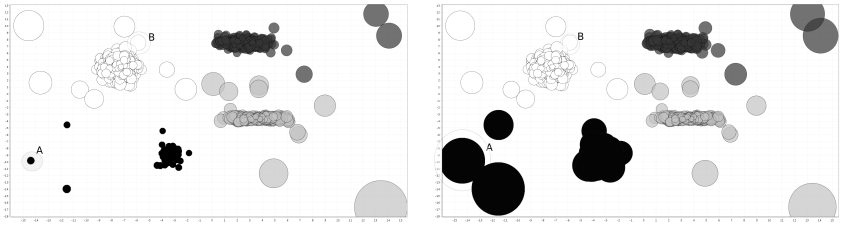


Figure 1: For p, the distance to the cluster center of C_1 is used for computing the CBLOF score. In this example C_1 and C_3 are identified as large clusters, while C_2 is considered to be a small cluster. The white points illustrate the cluster centers.

local as it does not take any local density into account - the amount of items in a cluster does not necessarily refer to its density.

Upon the conduction of the initial experiments, it has been observed that Equation 1 could lead to misleading results. Figure 2(a) shows an example of such a case. The black cluster is the only small cluster in this synthetic dataset. We have two points A and B; while it is obvious that A is more outlying than B, the score assignment (bubble size) shows otherwise. This can be explained by the fact that point B is multiplied by the size of the white colored cluster which is much larger than the size of the black cluster. The example shows that outlying points belonging to small clusters are discriminated against. Hence, we propose a variant to the algorithm *unweighted-CBLOF* shown by Equation 2.

$$unweighted\text{-}CBLOF(p) = \begin{cases} \min(d(p, C_j)) \text{ if } p \in SC \text{ where } C_j \in LC \\ d(p, C_i) \text{ if } p \in C_i \in LC \end{cases} \quad (2)$$



(a) CBLOF (b) unweighted-CBLOF

Figure 2: Comparing CBLOF with unweighted-CBLOF on a synthetic dataset. The size of the point indicates the outlier score. The gray level indicates to which of the four clusters the point belongs to.

Figure 2(b) shows the results of unweighted-CBLOF on the same artificial dataset. As expected point A has a higher outlier score than point B. What should be also noted, is that even the points deep inside the black cluster have a high outlier score as we consider the small clusters outlying.

In an attempt to apply the local density principle, we introduce *Local Density Cluster-Based Outlier Factor (LDCOF)*. Local density based anomaly detection approaches are popular as the anomaly score is normalized relative to the neighborhood. Moreover, the anomaly score has a natural threshold that would indicate whether the points are outlying or not.

The LDCOF score is defined as the distance to the nearest large cluster as illustrated in Figure 1 divided by the average distance to the cluster center of the elements in that large cluster. The intuition behind this is that when small clusters are considered outlying, the elements inside the small clusters are assigned to the nearest large cluster which becomes its local neighborhood. Thus the anomaly score is computed relative to that neighborhood.

$$distance_{avg}(C) = \frac{\sum_{i \in C} d(i, C)}{|C|} \tag{3}$$

$$LDCOF(p) = \begin{cases} \frac{\min(d(p, C_j))}{distance_{avg}(C_j)} \text{ if } p \in C_i \in SC \text{ where } C_j \in LC \\ \frac{d(p, C_i)}{distance_{avg}(C_i)} \text{ if } p \in C_i \in LC \end{cases} \tag{4}$$

The algorithms discussed above work on the output of any useful clustering algorithm. However, some clustering algorithms produce better results than others. Algorithms that take the number of clusters as a parameter, e.g. k-means and k-medoids are in favor compared to algorithms that determine the number of clusters automatically, for example X-means. It seems that overestimating the number of clusters performs in general better. Using more clusters is particularly useful in case of having non-spherical distributions where instances at the peripherals could falsely be identified as outliers. Additionally, the algorithm also considers small clusters as outlying, thus the excessive division into clusters helps in reducing false positives.

4 Optimizations

4.1 Handling Duplicates

Local k-nearest-neighbor based approaches can face some problems in case of having duplicates in the dataset. This arises as the density is inversely proportional to the distance. In case we have at least $k + 1$ duplicates of a certain data instance, the estimated density will be infinite. The solution that was proposed in [4] was utilized for these cases, which states that the distance to k^{th} neighbor would be calculated relative to the data with distinct spatial coordinates. Meaning that if we have $D = \{p_1, p_2, p_3, p_4\}$ where the coordinates of p_2 is the same as p_3 and $d(p_1, p_2) = d(p_1, p_3) \leq d(p_1, p_4)$, then the distance to the 2^{nd} nearest-neighbor would be $d(p_1, p_4)$.

Our implementation handles the duplicates as follows. The original dataset is processed by removing all the duplicates and assigning to each record in the

new dataset a corresponding weighting factor which is equal to the number of records with the same coordinates in the original dataset. The algorithms operate on the new dataset and finally the scores are mapped while producing the result set.

Besides solving the problem of duplicates, the preprocessing leads to another advantage: It can significantly reduce the number of records which reflects positively on the execution time.

4.2 Parallelization

The main bottle neck for the nearest-neighbor based algorithms is the need to compute the distances between each pair of data instances in the dataset. For a dataset of size n, this computation is typically done in n^2 operations, however it can be done in $\frac{(n-1)\cdot n}{2}$ if we take into account that the distance functions are symmetric.

Besides the computational complexity, also the memory consumption has to be minimal. Thus, our implementation does not store the complete distance matrix in memory but only a list of the best k-nearest-neighbors found so far. This list is then updated during the distance computation process if a closer neighbor is found. This reduces the memory consumption from $O(n^2)$ to $O(nk)$ dramatically making it possible to process large datasets. Additionally, the use of data structures containing non-primitive Java types was kept to a minimum.

In order to speed up the algorithms the code was parallelized for computing the k-nearest-neighbors. This can be done by one of the following methods. First we can compute the n^2 distances and avoid the need for any synchronization. The second option would be to compute only $\frac{(n-1)\cdot n}{2}$ distances, this would need some kind of synchronization because two threads could attempt to update the best-k-nearest-neighbors list of the same point at the same time. Synchronized blocks were used for synchronization since using Java ReentrantLock was significantly slower.

The comparison of the two proposed methods was not straight forward. This is due to the trade off between the time it takes to compute the distance function (which depends on the number of dimensions) and the waiting time of the threads. Figure 3 shows the results of running the two approaches on 100,000 randomly generated data instances with varying the dimensionality using 4 threads. Since the order of the dataset can affect the execution time, the average time of three runs with randomly shuffled input was taken.

It can be derived from Figure 3 that for smaller dimensions avoiding synchronization is better while for larger dimensions computing less distances is faster. This is because for higher dimensions the cost of computing the distances becomes more expensive than the waiting time of the threads. We defined a threshold of 32 for the dimensionality of the dataset. Datasets having lower dimensionality will be parallelized without synchronization, whereas

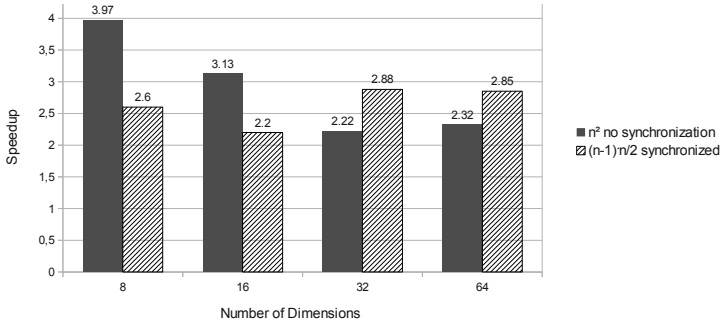


Figure 3: The nearest-neighbor search speed-up of the two proposed methods using 4 cores varying the dimensionality of the dataset. The first method avoids synchronization and the second (hatched) is using synchronized blocks.

synchronization is used for datasets with higher dimensionality.

5 Experiments

Experiments were performed on real world datasets from the UCI machine learning repository [11]. Basically, our evaluation has two main goals: First, it was used to verify that the algorithms have been implemented correctly by running them on datasets that were previously used in the literature. Second, we noticed that available literature today does not compare a sufficient amount of different algorithms. In this context, we compare the performance of up to eight different algorithms on global and local anomaly detection tasks.

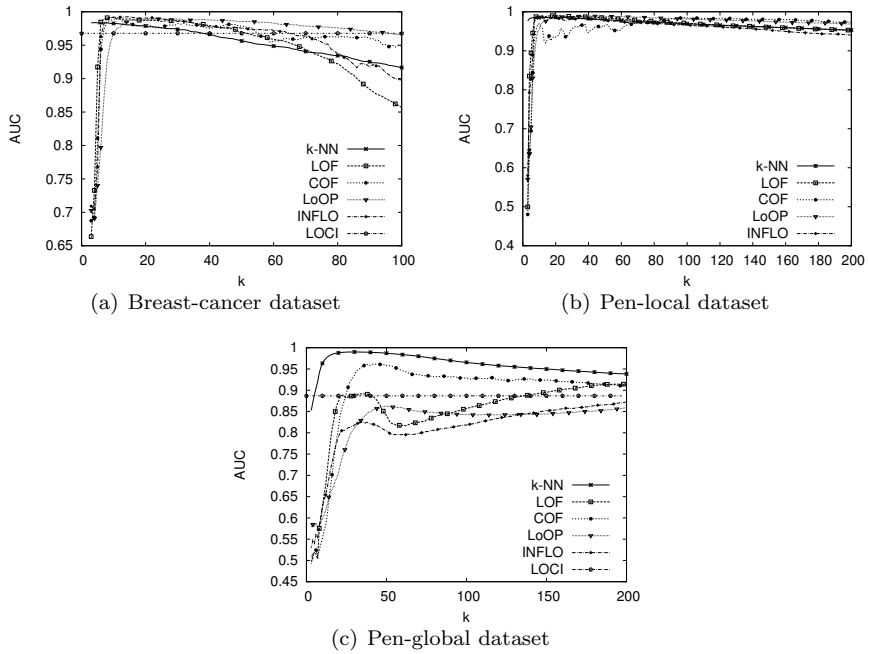
The first dataset used is the Breast Cancer Wisconsin (Diagnostic) dataset. The preprocessing applied is similar to what was performed in [7] where only the first 10 malignant records are kept. This dataset will be referred to as *breast-cancer* for the remainder of this paper.

The second dataset is the Pen-Based Recognition of Handwritten Text dataset[2]. The dataset consists of digits written by 45 different writers. Two different preprocessing steps are applied on this dataset. The first method is similar to the one performed in [7] where the digit 4 was chosen to be the anomalous class and only the first 10 records are kept in the dataset, resulting in a dataset of size 6724. This can be interpreted as a local anomaly detection problem since multiple classes with different densities may exist (referred to as *pen-local*). As a second preprocessing method, the digit 8 was chosen to be the normal class and thus all the remaining classes are sampled keeping only the first 10 digits of each class. This results in a smaller dataset of size 809, which can be interpreted as a global anomaly detection problem having one

[2]Only the training dataset was used

normal class and some outliers which are supposed to be further away from the normal class (referred to as *pen-global* in the following).

Results and Discussion



(a) Breast-cancer dataset

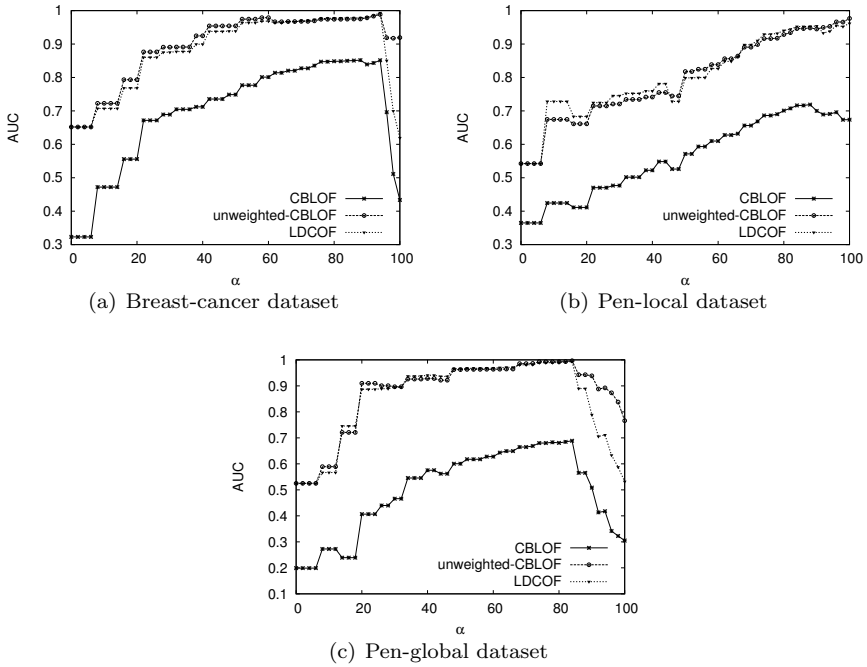
(b) Pen-local dataset

(c) Pen-global dataset

Figure 4: AUC for nearest-neighbor based algorithms for different k. It is shown that the choice of k strongly depends on the data and that local methods tend to fail on the global anomaly detection task (c).

For performance comparison, the receiver operating characteristic (ROC) was computed using the ranking of instances according to their anomaly scores. For simple comparison of the ROCs, the area under the curve (AUC) was computed further. Figure 4 shows AUC values of the nearest-neighbor algorithms on the different datasets. In 4(a) and 4(b) the local density based methods are superior to the global k-NN method. The performance of LOF and INFLO are very similar performing better at lower values of k, while for higher values of k, LoOP performs best. In Figure 4(a) and 4(c) LOCI was also plotted even though it does not vary with k. This was done in order to show how LOCI performs compared to the other approaches. For these datasets the performance of LOCI is worse than the other algorithms, however it can still be useful in case the user has no idea about the parameter settings of k. In Figure 4(c),

the global k-NN method performs best, followed by COF. It is an important finding, that global methods work fair enough on local problems but the the other way round is not true. Some of the local approaches fail significantly on the global anomaly detection problem - most likely due to the fact that the normal instances at the border of the normal class might be found as local outliers and score relatively high compared to the sparse global outliers.



(a) Breast-cancer dataset

(b) Pen-local dataset

(c) Pen-global dataset

Figure 5: The AUCs for clustering based algorithms reveal that our two proposed algorithms perform significantly better than the standard CBLOF. In contrast to the k in nearest-neighbor based methods, the choice of α is much more critical.

Figure 5 shows the effect of different values of α, the parameter affecting the partitioning into small and large clusters. The algorithms operate on the output of the k-means clustering algorithm with k set to 30. β was set to infinity such that the partitioning is only determined by the choice of α. Figures 5(a), 5(b) and 5(c) all show that both, unweighted-CBLOF and LDCOF are superior to CBLOF. Figures 5(a) and 5(c) show that the partitioning into small and large clusters is better than having no small clusters at all, which is at $\alpha = 100$. This could be due to the size of those two datasets which is much smaller than the size of pen-local. It is believed that all the implemented algorithms produce better results when the number of clusters (k for the k-means

algorithm) is overestimated.

Finally, a comparison of the performance of nearest-neighbor based algorithms with clustering based algorithms was performed. Table 1 shows the best AUC that can be obtained by varying the parameter settings. For the breast-cancer and the pen-local dataset the optimal k value for the nearest-neighbor based algorithms is 10 except for LoOP where k is 20. For the pen-global dataset the optimal k is 40. The optimal α for the clustering based algorithms for the breast-cancer dataset is 90, while for the pen-local dataset the optimal α for CBLOF is 90 and for the others it is 100. For the pen-global dataset, 80 is the optimal value for α. The performance of the nearest-neighbor algorithms and the clustering algorithms is comparable: The unweighted-CBLOF and LDCOF perform only slightly worse than the best nearest-neighbor based algorithms on the first two datasets. The same algorithms even slightly outperform the best nearest-neighbor based algorithm on the pen-global dataset. CBLOF performed worst on all datasets.

Table 1: Comparing AUC values for all algorithms using optimal parameter settings

Algorithm	Breast-cancer	Pen-local	Pen-global
k-NN	0.9826	0.9852	0.9892
LOF	0.9916	0.9878	0.8864
COF	0.9888	0.9688	0.9586
INFLO	0.9922	0.9875	0.8213
LoOP	0.9882	0.9864	0.8492
LOCI	0.9678	–[3]	0.8868
CBLOF	0.8389	0.7007	0.6808
unweighted-CBLOF	0.9743	0.9767	0.9923
LDCOF	0.9804	0.9617	0.9897

6 Conclusion

An *anomaly detection extension* for RapidMiner was implemented that contains the most well-known unsupervised anomaly detection algorithms. This extension will enable analysts to use those algorithms and integrate the operators into more complex processes easily. Also, the extension will enable researchers to perform further comparisons on the algorithms which would be beneficial as there is a general scarcity of those comparisons in the literature.

Our experiments show, that there is no overall best method for nearest-neighbor based algorithms and a good choice of the parameter k strongly depends on the data. Further it was shown that it is very important to know in advance if a global or local anomaly detection problem is to be addressed since local methods tend to fail on global tasks.

In the context of clustering based anomaly detection, unweighted-CBLOF and LDCOF are introduced as two new algorithms. The first results look

[3]Not computable due to too high memory requirements for this dataset using LOCI.

promising as they both outperform the existing CBLOF in our experiments. The observed behavior of both algorithms is very similar. However the scores of LDCOF are more readily interpretable and thus the method should be preferred. Similar to LOF, normal records have an anomaly score of approximately 1 while records having a score greater than 1 could be possible outliers.

Comparing nearest-neighbor and clustering based algorithms we found that nearest-neighbor based algorithms perform slightly better. However, if very large datasets have to be processed, one might choose clustering based algorithms since their complexity is usually lower, e.g. $O(nlog(n))$ using k-means in comparison of $O(n^2)$ for a nearest-neighbor search.

References

[1] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly Detection: A Survey. Technical report, University of Minnesota, 2007.

[2] Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. volume 2431 of *Lecture Notes in Computer Science*, pages 43–78. Springer, 2002.

[3] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proc. of the 2000 ACM SIGMOD*, pages "427–438", New York, NY, USA, 2000. ACM.

[4] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jrg Sander. LOF: Identifying Density-Based Local Outliers. In *Proc. of the 2000 ACM SIGMOD*, pages 93–104, USA, 2000. ACM.

[5] Jian Tang, Zhixiang Chen, Ada Fu, and David Cheung. Enhancing effectiveness of outlier detections for low density patterns. volume 2336 of *Lecture Notes in Computer Science*, pages 535–548. Springer, 2002.

[6] Wen Jin, Anthony Tung, Jiawei Han, and Wei Wang. Ranking outliers using symmetric neighborhood relationship. volume 3918 of *Lecture Notes in Computer Science*, pages 577–593. Springer, 2006.

[7] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Loop: local outlier probabilities. In *Proc. of CIKM '09*, pages 1649–1652, USA, 2009. ACM.

[8] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B. Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. *Data Engineering, International Conference on*, page 315, 2003.

[9] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641 – 1650, 2003.

[10] Moh'd Belal Al-Zoubi. An Effective Clustering-Based Approach for Outlier Detection. *European J. Scientific Research*, 28(2):310–316, 2009.

[11] A. Frank and A. Asuncion. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2010.