International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:9, 2007

Class Outliers Mining: Distance-Based Approach

Nabil M. Hewahi, Motaz K. Saad

Abstract**—**In large datasets, identifying exceptional or rare cases with respect to a group of similar cases is considered very significant problem. The traditional problem *(Outlier Mining)* is to find exception or rare cases in a dataset irrespective of the class label of these cases, they are considered rare events with respect to the whole dataset. In this research, we pose the problem that is *Class Outliers Mining* and a method to find out those outliers. The general definition of this problem is "*given a set of observations with class labels, find those that arouse suspicions, taking into account the class labels".* We introduce a novel definition of Outlier that is *Class Outlier*, and propose the *Class Outlier Factor (COF)* which measures the degree of being a *Class Outlier* for a data object. Our work includes a proposal of a new algorithm towards mining of the *Class Outliers*, presenting experimental results applied on various domains of real world datasets and finally a comparison study with other related methods is performed.

Keywords**—**Class Outliers, Distance-Based Approach, Outliers Mining.

I. Introduction

OUTLIERS mining is the problem of detecting rare events, deviant objects, and exceptions. Outliers mining is an important data mining issue in knowledge discovery, it has attracted increasing interests in recent years. Recently, researchers have begun focusing on this problem and have attempted to apply algorithms for finding outliers to tasks such as fraud detection [6], identifying computer network intrusions [8, 22], data cleaning [27], detecting employers with poor injury histories [18], and in other several problem domains (e.g., surveillance and auditing, stock market analysis, health monitoring systems, insurance, banking and telecommunication ..., etc). Methods for finding such outliers in large datasets are drawing increasing attention [2, 3, 10, 15, 16, 18, 19, 20, 21, 26].

A. Outlier Definition

An Outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism [11]. It is a data object that does not comply with the general behavior of the data, it can be considered as noise (*One person's noise could be another person's signal*) or exception, which is quite useful in rare events analysis [10].

Some of the very well known methods for outliers detection are statistical based (distribution based) [3, 11, 26], clustering [2, 9, 17], depth based [16], distance based [19], density based [7], and model based (Neural Networks) [12].

In section B, we shall discuss in details the distance-based method due to its relation to our approach.

B. Distance-Based Approach

There are two main methods that can be classified under distance-based approach.

1) K Nearest Neighbors (KNN)

Defining outliers by their distance to neighboring examples is a popular approach to finding unusual examples in a dataset. Recently, much work has been conducted with the goal of finding fast algorithms for this task [18, 20, 21].

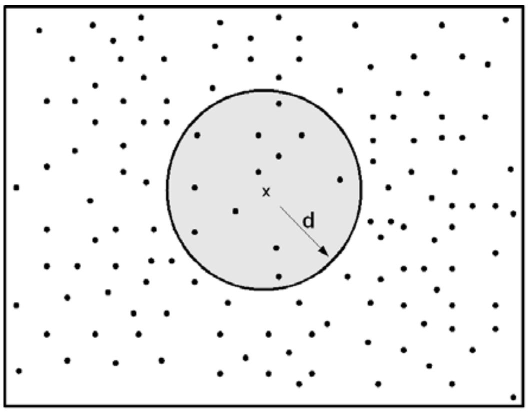

Fig. 1: Distance-Based Approach: *K* Nearest Neighbors

A popular method of identifying outliers is by examining the distance to an example's nearest neighbors [1, 17, 18, 25] as shown in Figure 1. In this approach, one looks at the local neighborhood of points for an example typically defined by the *K* nearest examples (also known as neighbors). If the neighboring points are relatively close, then the example is considered normal; if the neighboring points are far away, then the example is considered unusual. The advantages of distance-based outliers are that, no explicit distribution needs to be defined to determine unusualness, and it can be applied to any feature space for which we can define a distance measure. Below are some of distance-based outlier definitions:

Nabile M. Hewahi is with Department of Computer Science, Islamic University of Gaza, Palestine (e-mail: nhewahi@iugaza.edu).
Motaz K. Saad is with Department of Computer Science, Islamic University of Gaza, Palestine (e-mail: msaad@iugaza.edu).

1. Outliers are the examples for which there are fewer than *p* other examples within distance *d* [17, 18].
2. Outliers are the top *n* examples whose distance to the K^{th} nearest neighbor is greatest [25].
3. Outliers are the top *n* examples whose average distance to the *K* nearest neighbors is greatest [8, 18].

There are several minor differences between these definitions as shown in Figure 2. The first definition does not provide a ranking and requires specifying a distance parameter *d*. Ramaswamy et al. [25] argue that this parameter could be difficult to determine and may involve trial and error to guess an appropriate value. The second definition only considers the distance to the K^{th} neighbor and ignores information about closer points. Finally, the last definition accounts for the distance to each neighbor but it is slower to calculate than definition 1 or 2. However, all of these definitions are based on a nearest neighbors density estimate to determine the points in low probability regions which are considered outliers.

Researchers have tried a variety of approaches to find these outliers efficiently. The simplest are those using nested loops [17, 18, 25]. In the basic version, one compares each example with every other example to determine its *K* nearest neighbors. Given the neighbors for each example in the dataset, simply select the top *n* candidates according to the outlier definition.

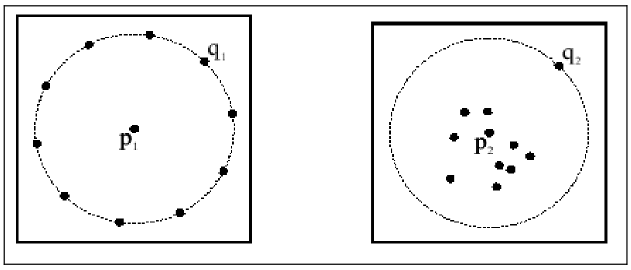

Fig. 2: The difference between Distance-based Outliers definitions. Both of p_1 and p_2 have 10 neighbors, in some approaches p_1 and p_2 have the same outlierness ranking and in other approaches p_1 has larger outlierness ranking than p_2

This approach has quadratic complexity as we must make all pair wise distance computations between examples. Although distance is an effective non-parametric approach to detecting outliers, the drawback is the amount of computation time required. Straightforward algorithms, such as those based on nested loops, typically require $O(N^2)$ distance computations. This quadratic scaling means that it will be very difficult to mine outliers as we tackle increasingly larger datasets. This is a major problem for many real databases where there are often millions of records. To overcome this problem, some new approaches have been developed to enhance the complexity to become semi-linear [4].

2) Density Based

This was proposed by Breunig, et al. [7]. It relies on the *Local Outlier Factor* (*LOF*), which is the average of the ratios of the density of example *p* and the density of its nearest neighbors. *LOF* depends on the local density of its neighborhood. The neighborhood is defined by the distance to the *MinPts-th* nearest neighbor, where *MinPts* is the minimum number of points of the nearest neighbors.

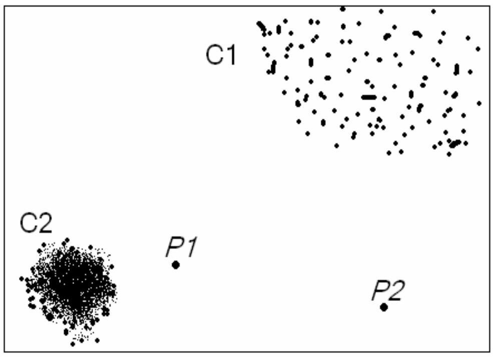

Fig. 3: The Concept of Density Based Outlier.

Figure 3 shows the concept of density based outliers and the difference between the nearest neighbors approach and the density based approach, where in the nearest neighbors (*NN*) approach, p_2 is not considered as outlier, while *LOF* approach find both p_1 and p_2 as outliers. In typical use, points with a high *LOF* are flagged as outliers. The process steps as following:

- Compute density of local neighborhood for each point.
- Compute *LOF*.
- Choose instances with larger *LOF* as outliers.

Density Based approach was proposed primarily to deal with the local density problems of the distance based method. However, selecting *MinPts* is non-trivial. In order to detect outlying clusters, *MinPts* has to be as large as the size of these clusters.

II. RESEARCH OBJECTIVES

From the previous discussion, we notice that all the mentioned approaches do not consider the class labels of the dataset, rather, they focus on the "*observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism*". This means all the previous methods are devoted on the overall dataset without looking closely to each class label separately.

Obviously, in the *K* Nearest Neighbor systems, it is expected that the instances in *KNN* are to be identified in the same class label. However, this is not always true. It is reasonable to take those instances whose class label is different from that of the majority of the *KNN* as a *Class Outliers* with consideration also to other factors. More details will be discussed later.

To show the significance of *Class Outliers*, let us notice the following examples. Consider the problem of finding a voter from democrat party that behaves or acts like republicans, in other words, what is the percentage of Democrats that act like (have similar ideas) Republicans and vice versa. Table 1 is the outcome of our proposed approach applied on house-vote-84 dataset [5] (to be explained later) and stated here only to show the importance of the *class outliers*. Numbers 1-16 in table 1 indicate the issue numbers to be voted about. Inst. # in the table indicates the instance number in the dataset. The table shows that the class label of instance #407 is different from that of its neighbors although it is very similar to them. More information about the domain, dataset, and experimental details are presented in Implementation and Experimental Results section.

TABLE I
THE NEAREST NEIGHBORS OF INSTANCE #407 OF HOUSE-VOTE-84 DATASET

Att.# / Inst#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Class
407	n	n	n	y	y	y	n	n	n	n	y	y	y	y	n	n	democrat
306	n	n	n	y	y	y	n	n	n	n	n	y	y	y	n	n	republican
83	n	n	n	y	y	y	n	n	n	n	n	y	y	y	n	n	republican
87	n	n	n	y	y	y	n	n	n	n	n	y	y	y	n	n	republican
303	n	n	n	y	y	y	n	n	n	n	n	y	y	y	n	n	republican
119	n	n	n	y	y	y	n	n	n	n	n	y	y	y	n	n	republican
339	y	n	n	y	y	y	n	n	n	n	y	y	y	y	n	n	republican

TABLE II
THE INFORMATION GAIN OF HOUSE-VOTE-84 DATASET'S FEATURES

Issue #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Information Gain	0.13	0	0.43	0.74	0.42	0.15	0.2	0.34	0.31	0.01	0.11	0.37	0.23	0.34	0.22	0.1

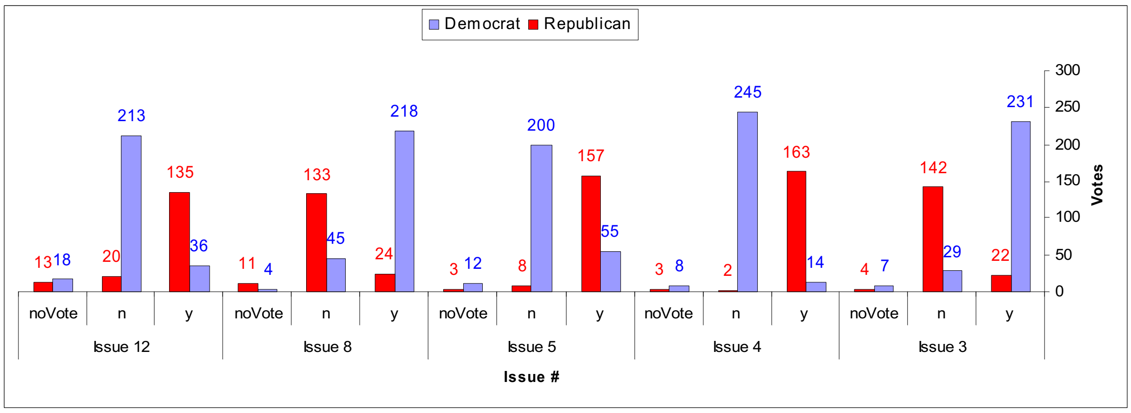

Fig. 4: Party Behavior of house-vote-84 dataset

TABLE III
THE 7 NEAREST NEIGHBORS OF THE INSTANCE #69 OF HEART-STATLOG DATASET

Att.# / Inst#	1	2	3	4	5	6	7	8	9	10	11	12	13	Class
69	47	1	3	108	243	0	0	152	0	0	1	0	3	present
62	44	1	3	120	226	0	0	169	0	0	1	0	3	absent
150	41	1	3	112	250	0	0	179	0	0	1	0	3	absent
179	50	1	3	129	196	0	0	163	0	0	1	0	3	absent
38	42	1	3	130	180	0	0	150	0	0	1	0	3	absent
253	51	1	3	110	175	0	0	123	0	.6	1	0	3	absent
23	47	1	3	112	204	0	0	143	0	.1	1	0	3	absent

To understand why the instance #407 (Democrat) is close to a set of Republican instances, an analysis has been performed on the whole dataset and found that the majority of democrats vote in issues 3, 4, 5, 8 and 12 are anti-republican, however, instance #407 in all previous issues voted pro-republican. Figure 4 depicts the party behavior in the issues 3, 4, 5, 8 and 12. For example, minority of Republicans vote “yes” for issue 3, although the majority of Democrats vote “yes”. Similarly it is the case for the same issue in voting “no” with majority of Republicans and minority of Democrats. Table 2 shows the information gain of house-vote-84 dataset's features, the table clarify that features 4, 3, 5, 12, and 8 have the highest information gain respectively, and they are more important than other features for the class label.

In a medical/biological domains, consider the problem of finding the exceptional case (or cases) of a group of similar cases, where the class label is a medical diagnoses (like “absent” and “present” in heart-statlog dataset [5]). The question is why the class label of one of the *KNN* is “present” while the class label of its *KNN* is “absent”. Table 3 (Att.# in the table means attribute #) shows a case where most of the inputs for instance #69 are mostly similar to its seven nearest neighbors but its class is different. Instance #69 is considered to be *class outlier*. Giving this table, we are not trying to give explanations on how this case medically happened, but it remains as an interesting question which has to be answered by doctors.

Class outliers have advantages and applications like data preprocessing and cleaning, credit card fraud detection, network intrusion detection, stock market analysis, health care and monitoring, ...etc., (in general problem of detecting rare events, deviant objects, and exceptions), furthermore, *class outliers* have very promising potential advantages, applications, and new future research directions.

To the best of our knowledge, the problem of "*given a set of observations with class labels, find those that arouse suspicions, taking into account the class labels"* has only been explicitly considered in [13, 14, 24]. The proposed methods are *Semantic outlier* [13], *Cross-Outlier* [24], *Class Outlier* [14]. It is obvious that other outliers detection methods can not detect such type of outlierness.

He, et al. [13] tried to find meaningful outliers that called *Semantic Outlier Factor (SOF).* The approach is based on applying a clustering algorithm on a dataset with a class label, it is expected that the instances in every output cluster are to be identified with the same class label. However, this is not always true. Figure 5 illustrates the concept of *SOF* for a dataset with two class labels *x*, and *y*. It is reasonable to take those instances whose class label is different from that of the majority of the cluster as *semantic outlier*. The *Semantic outlier* definition is a data point, which behaves differently with other data points in the same class, while looks normal with respect to data points in another class.

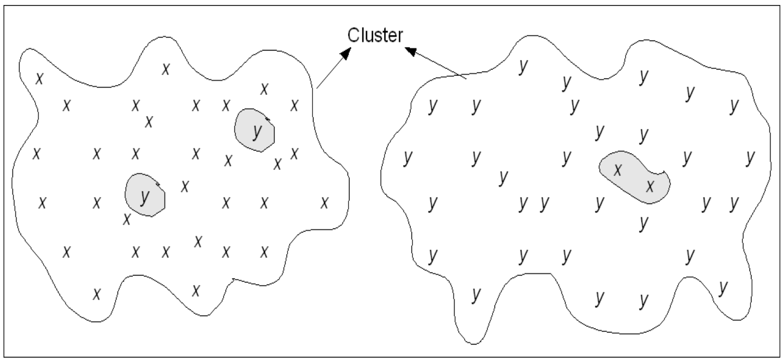

Fig. 5: The Concept of *SOF*. Instances in dark areas are considered outliers

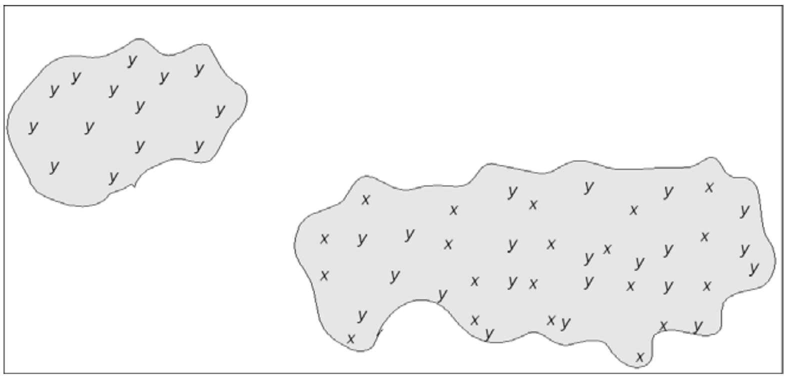

Fig. 6: The concept *Cross-Outlier*: ys group is considered suspicious

SOF considers the following to rank an instance *T*:

- The probability of the class label of the instance *T* which belongs to a certain cluster with respect to the class labels of the instances in the same cluster.
- The similarity between the instance *T* and the instances in the same class.

Papadimitriou and Faloutsos [24] tried to solve the problem: Given two sets (or classes) of objects, find those which deviate with respect to the other set. Those points are called *Cross-outlier,* and the problem is identified by *Cross-outlier detection.* In this case we have a primary set *P* in which we want to discover *cross-outliers* with respect to a reference set *R* (detecting outlying observations: discover points $p \in P$ that "arouse suspicions" with respect to points $r \in R$). The proposed solution is to use a statistically intuitive criterion for outlier flagging (the local neighborhood size differ more than three standard deviation from the local average), with no magic cut-offs. Papadimitriou and Faloutsos [24] considered that some single class approaches may be modified to deal with multiple classes, but the task is non-trivial. The general problem is open and provides promising future research directions. The authors generally considered the existing approaches for the single-set problem, are not immediately extensible to cross-outlier detection. Also several outlier definitions themselves can not be extended.

Figure 6 depicts the concept of *cross-outlier*, the Figure shows the data objects are grouped in two regions, one group contains data objects belong to the class labels *x* and *y*, and the other group contains data objects belong to class *y.* Based on the definition of *Cross-Outlier*, the two datasets (data objects with *x* class and data objects with *y* class) are correlated, this implies that the group which contains only data objects of class *y* is suspicious. The perspective of [24] about *class outliers* is different from ours as we shall see.

He, et al. [14] tried to find a general framework to contributions presented in [24, 13] by proposing a practical solution and extending existing outlier detection algorithms. The generalization does not consider only outliers that deviate with respect to their own class, but also outliers that deviate with respect to other classes. In addition, potential applications of customer relationship management (*CRM*) are introduced.

In this research we propose a new method for mining *class outliers* based on distance-based approach and nearest neighbors by introducing the Concept of *Class Outlier Factor (COF)* which represents the degree of being a *class outlier*. Also we try to overcome some limitations of the related methods.

The main limitations of the previously proposed methods are that they do not handle numeric or mixed dataset. Moreover, in [13, 14], a clustering as a pre-process has to be performed. Surely, this would increase the computational complexity. In addition to that, the approach used in [13, 14] is based on the probability of the occurrence of the outlier within certain cluster to specify the rank of the outlier. This might produce the same rank for very far/close outliers in the cluster. In [13, 14], the proposed approach does not handle datasets with more than two classes, whereas, our proposed method takes care of this problem.

The main contributions of our research are the following:

- Proposing Distance-Based *Class Outlier* definition and introducing the Concept of *COF.*
- Proposing and implementing ***C**lass **O**utliers: **D**istance-**B**ased (CODB)* Algorithm for mining *Class Outliers*.
- Presenting experimental results of the *CODB* algorithm tested on various real world datasets.
- Performing a comparison study with results of other related methods presented in [13,14].

III. The Proposed Scheme

A. Definitions and terms

Before going into the details of the proposed approach, we shall give the following definitions.

1) Distance (Similarity) Function

Given a dataset $D = \{t_1, t_2, t_3, ..., t_n\}$ of tuples where each tuple $t_i = <t_{i1}, t_{i2}, t_{i3}, ..., t_{im}, C_i>$ contains m attributes and the class label C_i, the similarity function is based on the *Euclidean* Distance [10] between two data tuples, $X = <x_1, x_2, x_3, ..., x_m>$ and $Y = <y_1, y_2, y_3, ..., y_m>$ (excluding the class labels):

$$d_2(X,Y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} \tag{1}$$

And the similarity function is based on *Manhattan* distance:

$$d_1(X,Y) = \sqrt{\sum_{i=1}^{m}|x_i - y_i|} \tag{2}$$

For numeric attributes, similarity is determined by normalizing attribute values. An attribute is normalized by scaling its values so that they fall within a small specific range, such 0.0 to 1.0 [10]. Normalization is particularly useful for classification algorithms including neural networks, or distance measurements such as nearest neighbors systems, classification and clustering. Normalization speeds up the learning phase in training and prevents attributes with initially large ranges (e.g., income) from outweighing attributes with initially smaller ranges (e.g., binary attributes). There are many methods for data normalization: *min-max normalization, z-score normalization,* and *normalization by decimal scaling* [10].

Symbolic (nominal) features are more problematic as they do not fit in the *Euclidean* feature space model. To overcome this problem, similarity between symbolic features is determined by counting the matching features. This is a much weaker function as there may be several concepts based on entirely different features, all of which match the current example to the same degree. For domains containing a mixture of numeric and symbolic features the *Euclidean* distance function is adopted, with the distance between two symbolic values trivialized to zero if the features are the same, and one if they are not. This mismatch between *Euclidean* feature space and symbolic features means that, pure nearest neighbor systems usually perform better in numeric domains than in symbolic ones.

2) K Nearest Neighbors

For any positive integer *K*, the *K*-Nearest Neighbors of a tuple t_i are the *K* closest tuples in the dataset, as shown in Figure 7.

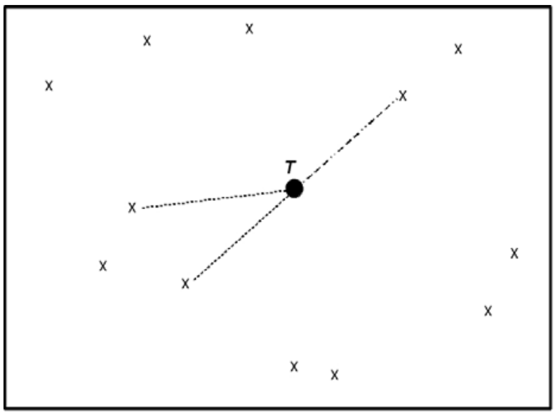

Fig. 7: the K Nearest Neighbors of the instance *T* (*K* = 3)

3) PCL

PCL(*T, K*): The Probability of the class label of the instance *T* with respect to the class labels of its *K* Nearest Neighbors

For example, suppose we are working with 7 nearest neighbors of an instance *T* (including itself) on a dataset with two class labels *x* and *y*, where 5 of these neighbors have the class label *x*, and 2 have the class label *y* as shown in Figure 8. The instance *T* has the class label *y,* which means the *PCL* of the instance *T* (The probability of the class label *y* to the other class labels of the nearest neighbors) is 2/7.

.

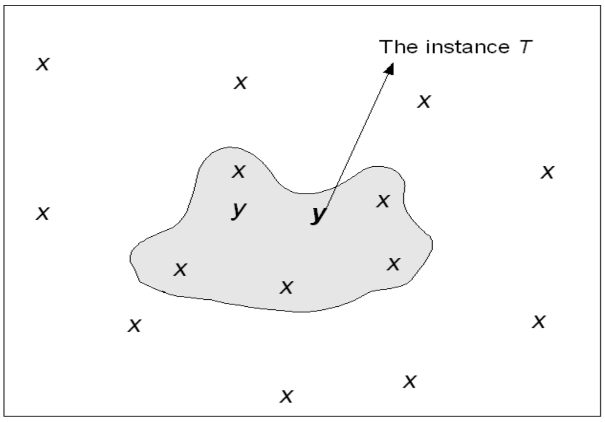

Fig. 8: The probability of the class label of the instance *T* with respect to the class labels of its nearest neighbors

4) Deviation

Given a subset *DCL = {t_1, t_2, t_3, ..., t_h}* of a dataset *D = {t_1, t_2, t_3, ..., t_n}*. Where *h* is the number of instances in *DCL* and *n* is the number of instances of *D*. Given the instance *T, DCL* contains all the instances that have the similar class label of that of the instance *T*.

The *Deviation* of *T* is how much the instance *T* deviates from *DCL* subset. The *Deviation* is computed by summing the distance between the instance *T* and every instance in *DCL*. *Dev(T)* is defined as:

$$Dev(T) = \sum_{i=1}^{h} d(T, t_i), \quad Where \quad t_i \in DCL \tag{3}$$

Figure 9 demonstrate the importance of *Deviation* Factor, although the *PCL* of y_1 and y_2 are the same (*K* = 5), but y_2 deviates more than y_1 from instances with y class. This will reflect the decision of *class outlier* in favor of y_2.

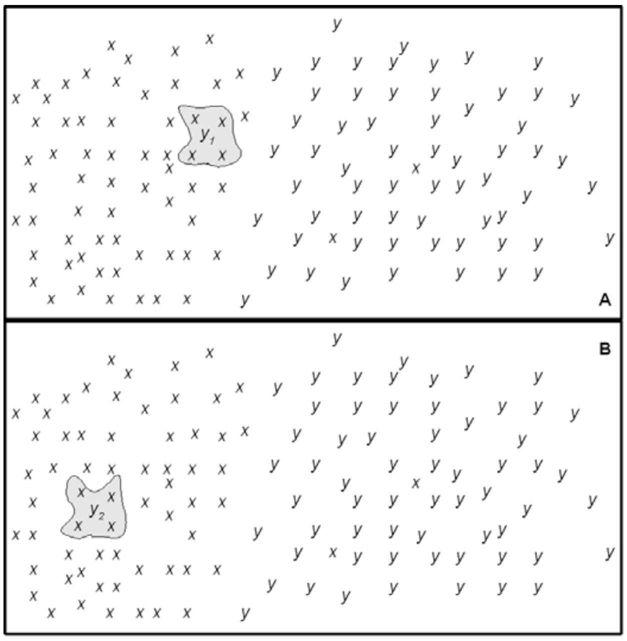

Fig. 9: Deviation: y_2 deviates from instances with *y* class more than y_1

5) K-Distance (The Density Factor)

K-Distance(T) is the Distance between the instance *T* and its *K* nearest neighbors, i.e. how much the *K* nearest neighbors instances are close to the instance *T. KDist(T)* is defined as:

$$KDist(T) = \sum_{i=1}^{K} d(T, t_i) \tag{4}$$

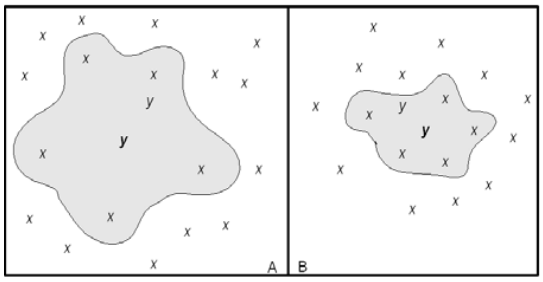

Fig. 10: K-Distance (Density Factor)

Figure 10 shows the *KNN* of the instance *y* (*the bold y*). In Figure 10.B the *KNN* of the instance bold *y* are much closer (higher density) to it than the *KNN* of the instance bold *y* in Figure 10.A. Although the *PCL(T, K)* for both instances in Figure 10.A and Figure 10.B are equal and is 2/7, the instance bold *y* in Figure 10.B is considered to be more *class outlier* than the instance bold *y* in Figure 10.A.

6) Class Outlier

Class Outliers are the top *N* instances which satisfy the following:

1. The *K-Distance* to its *K* nearest neighbors is the least.
2. Its *Deviation* is the greatest.
3. Has different class label form that of its *K* nearest neighbors.

7) Class Outlier Factor (COF)

The *Class Outlier Factor* of the instance *T* is the degree of being *Class Outlier*. *The Class Outlier Factor* of the instance *T* is defined as:

$$COF(T) = K * PCL(T,K) + \alpha * \frac{1}{Dev(T)} + \beta * KDist(T) \quad (5)$$

Where *PCL(T, K)* is described in section III.A.3, *Dev(T)* and *KDist(T)* are described in formulas 3 and 4 respectively.

As shown above, we scaled *PCL(T, K)* from [*1/K,1*] to [*1,K*] by multiplying it by *K*. α and β factors are to control the importance and the effects of *Deviation* and *K-Distance.* The values of α and β are $0 \le \alpha \le M$ and $0 \le \beta \le 1$ respectively, where *M* is a changeable value based on the application domain and the initial experimental results. If the maximum *Deviation* of the instance in the initial experimental results is in hundreds, then the optimum value for α is 100, and if the it is in tens, then the optimum value for α is 10 and so on. The optimum value of β is 0.1 if the maximum *K-Distance* in the initial experimental results is in tens, and 0.01 if the maximum *K-Distance* in the initial experimental results is in hundreds, and so on. The main goal of scaling *PCL*, α and β factors is to obtain the *COF* value in the format *X.YYYY* where *X* reflects the scaled *PCL* and *YYYY* reflects the *Deviation* and *KDist* factors. We consider *PCL* as the most important factor to take a decision regarding the Class Outlierness. α and β factors are considered to make a trade-off between the importance of *Deviation* and *KDist*. The proposed ranges for α and β are chosen in a way to maintain the trade-off.

The optimal value of *K* is determined by trial and error technique. Of course, there are many factors affecting the optimal value, for example, dataset size and number of classes are very important factors that affect choosing the value of *K*. Keeping a very high value of *K* would necessarily mean we are not sure in localized regions of the search space (e.g., instances from the other classes would enter the search space (*KNN* region)) and this might result in wrong estimation for *PCL*. On the other hand, keeping an extremely low value of *K* means *KNN* is not well utilized and will give wrong impression about the importance of *PCL*. Odd values of *K* would make more sense because we would like to have a clear bias value for the *PCL*.

B. The Proposed Algorithm (CODB)

In this section we present the proposed algorithm. We call our proposed algorithm *"**C**LASS **O**UTLIERS: **D**ISTANCE-**B**ASED"* (*CODB* Algorithm). Figure 11.A shows the pseudo code of *CODB* algorithm, Figure 11.B presents the *Rank Procedure* algorithm which is called by *CODB,* and Figure 12 depicts the *CODB* algorithm flowchart.

```
CODB Algorithm
Input:
    D = {t_1, t_2, t_3, ..., t_n}    /*Dataset*/
    n    /*Dataset size*/
    α    /*Alpha Factor*/
    β    /*Beta Factor*/
    K    /*Number of Nearest Neighbors*/
    N    /*Number of Top Class Outliers*/
Output:
    Top N Class Outliers and their COF (Class Outlier Factor) value
Process:
    list = new List(N); /*Initialize empty set (empty list) with size N */
    COF = 0; /*Initialize Class Outlier Factor*/
    /*Process each instance in the dataset to rank it with COF, and keep only the top N
    Class Outliers in the list */
    for i = 1 to n {
            COF = Rank(Instance(i), K, α, β ); // compute COF for the instance i
            if list.Size() < N then  // if the list is not full
                    list.Add(Instance(i)); // the add the instance i to the list
            else { /* Keep only top N Class Outliers (that have smaller COF value)*/
                    if COF < list.GetMax() then {
                            list.RemoveMax(); /*Remove instance with highest
                            COF value*/
                            list.Add(Instance(i)); /*Keep only top N Class Outliers
                            (with smaller COF value)*/
                            }
                    }
            }
    Print(list); /*print out the top N class outliers list*/
```

A

```
Procedure: Rank(Instance(i), K, α, β )
Input:
    Instance(i)    /* The instance i*/
    K    /*Number of Nearest Neighbors*/
    α    /*Alpha Factor*/
    β    /*Beta Factor*/
Output:
    COF  /*Class Outlier Factor (Degree of Outlierness: Smaller
    → Top Class Outliers)*/
Process:
    /*Initialize PCL */
    PCL = 0;
    /*Initialize Deviation */
    Deviation  = 0;
    /*Initialize KDist */
    KDist = 0;
    /* compute the PCL value of the instance T */
    PCL = PCL(T, K);
    /* compute the Deviation of the instance T */
    Deviation = Deviation(T);
    /* compute the KDist value of the instance T */
    KDist = KDist(T);
    /* compute the COF value of the instance T */
    COF = K * PCL + (α / Deviation) + (β * KDist);
    return COF;
```

B

Fig. 11: A.: The *CODB* Algorithm. B.: The *Rank Procedure* Algorithm

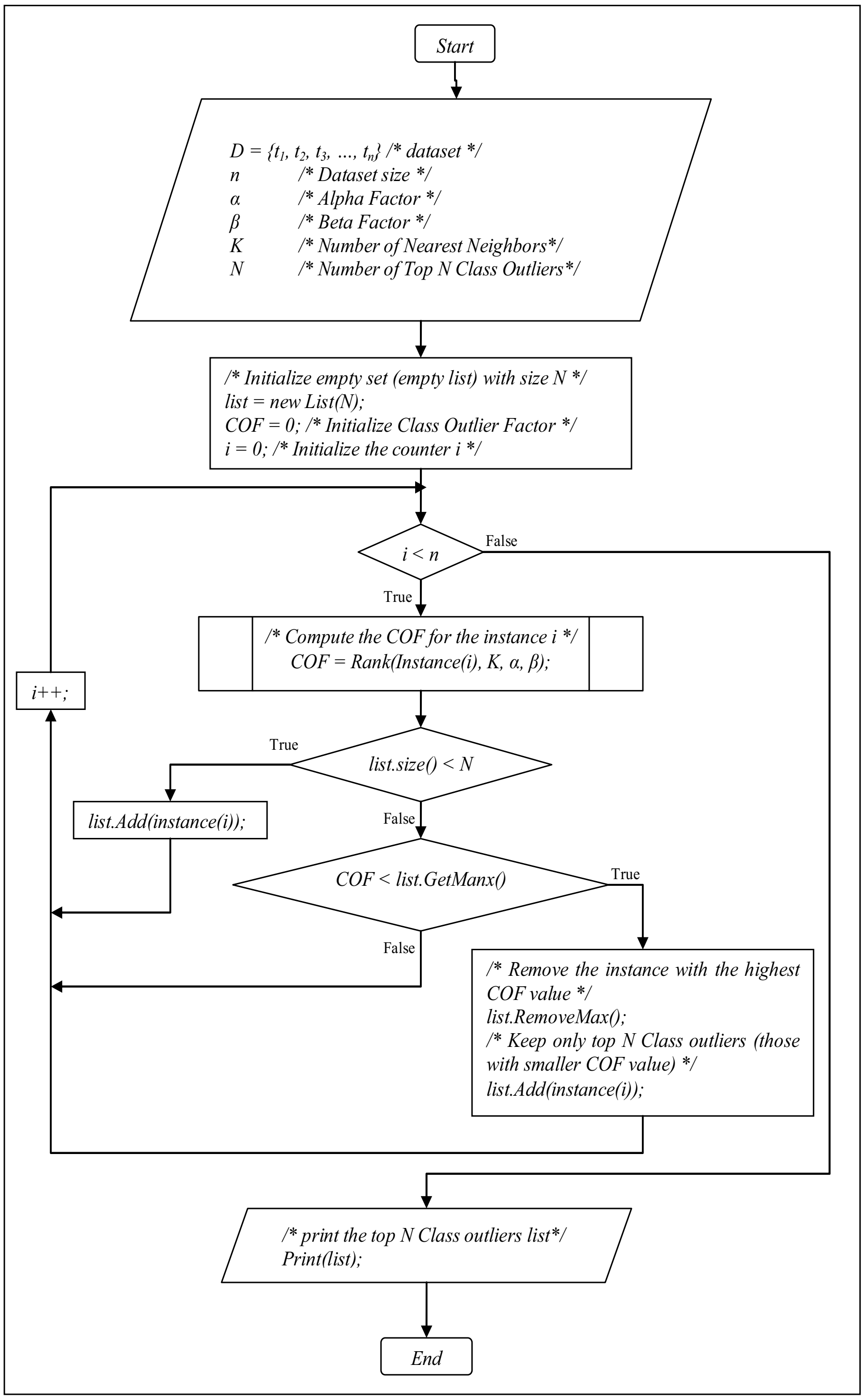
Start
$D = \{t_1, t_2, t_3, \ldots, t_n\}$ /* dataset */
n /* Dataset size */
α /* Alpha Factor */
β /* Beta Factor */
K /* Number of Nearest Neighbors*/
N /* Number of Top N Class Outliers*/
/* Initialize empty set (empty list) with size N */
list = new List(N);
COF = 0; /* Initialize Class Outlier Factor */
i = 0; /* Initialize the counter i */
$i < n$
False
True
/* Compute the COF for the instance i */
COF = Rank(Instance(i), K, α, β);
i++;
True
list.size() < N
list.Add(instance(i));
False
COF < list.GetManx()
True
False
/* Remove the instance with the highest COF value */
list.RemoveMax();
/* Keep only top N Class outliers (those with smaller COF value) */
list.Add(instance(i));
/* print the top N Class outliers list*/
Print(list);
End

Fig. 12: The *CODB* algorithm flowchart

The main concept of *CODB* is to rank each instance in the dataset *D*. This is done by calling the Rank procedure after providing the *CODB* with all the necessary data such as the value of *α*, *β* and *K*. The Rank finds out the rank of each instance using the formula 5 and gives back the rank to *CODB*. The *CODB* maintains a list of only the instances of the top *N* class outliers. The less is the value of *COF* of an instance, the higher is the priority of the instance to be a class outlier.

IV. Implementation and Experimental Results

The *CODB* algorithm has been applied on five different real world datasets. All the datasets are publicly available at the *UCI* machine learning repository [5]. The datasets are chosen from various domains that might have single or mixed data types and with two or more class labels. This variation is being tested on our proposed algorithm to show its capabilities.

WEKA framework [28] has been used to apply certain functions related to our proposed algorithm. Other functions have been developed and implemented using Java.

Throughout the experiments, missing values in the datasets can be either handled by instance removal of the corresponding missing values, or the values are replaced by the mean in case of continues attributes and the mode in case of nominal attributes.

The datasets that have been tested by *CODB* are listed below:

- House-vote-84 dataset: Nominal, 2 class labels.
- Vehicle dataset: Continues, 4 class labels.
- Hepatitis dataset: Mixed, 2 class labels.
- Heart-statlog dataset: Mixed, 2 class labels.
- Credit approval (credit-a) dataset: good mix of attributes (continuous, nominal with small numbers of values, and nominal with larger numbers of values), 2 class labels.

Only two experiments of the five experiments are presented in details, where the other three experiments are presented in brief.

A. Experiment I (house-vote-84 dataset)

The dataset of house-vote-84 [5] (1984 United States Congressional Voting Records Database) includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes. There are 16 attributes + class name = 17 attributes, all boolean valued with Yes (denoted as “y”) and No (denoted as “n”), there are 435 instances belonging to two classes, i.e. democrats or republicans. The class distribution is 267 democrats, 168 republicans (61.38% Democrats, 38.62% Republicans). Missing attribute values are denoted by "?" in the original dataset. It is important to recognize that "?" in this database does not mean that the value of the attribute is unknown, it means simply, that the value is neither “yea" nor "nay" [5]. So we replaced all “?” by “noVote” value to represent the real position of the voter, and to avoid handling it as a missed value in the experiments to get more reality. The following inputs are provided to the implemented algorithm:

K = 7, *Top N COF* = 20, *α* = 100, *β* = 0.1, *Remove Instance with Missing Values:* false, *Distance type: Euclidean* Distance, *Replace Missing Values:* false.

Table 4 shows the top 20 *class outliers* whereas table 5 shows the distance of rank 1, 2, and 11 *class outlier* instances from their *K* nearest neighbors (7 nearest neighbors). Table 4 shows that instance #407 is at the top where its *PCL* is 1/7 and its *COF* is the least (i.e., 1.71158). Comparing instance #407 with instance #375 (rank 2), we notice that the *Deviation* of instance #407 is more than the *Deviation* of instance #375, but the *KDist* of instance #407 is less than that of instance #375. This indicates that instance #407 deviates from its class in a very dense area of other class. As we go down along with the table, we notice that the value of *PCL* is increasing, which reflects the reality that our main factor is the *PCL*. The distinguishing between any instances with the same *PCL* value, is the values of *Deviation* and *KDist*.

TABLE IV
THE TOP 20 CLASS OUTLIERS OF HOUSE-VOTE-84 DATASET

#	Inst. #	PCL	Dev	KDist	#	Inst. #	PCL	Dev	KDist
1	407	1	896.24	6.0	11	176	2	519.94	8.49
		COF: 1.71158					COF: 3.04086		
2	375	1	881.78	8.07	12	384	2	832.95	9.34
		COF: 1.92051					COF: 3.0543		
3	388	1	857.35	8.07	13	365	2	836.65	9.44
		COF: 1.92375					COF: 3.0634		
4	161	1	819.35	8.49	14	6	2	849.33	9.76
		COF: 1.97058					COF: 3.0934		
5	267	1	523.66	8.49	15	355	2	524.28	9.12
		COF: 2.03949					COF: 3.10283		
6	71	1	535.52	9.44	16	164	2	845.19	10.07
		COF: 2.13061					COF: 3.12576		
7	77	1	799.64	10.39	17	402	2	480.79	10.93
		COF: 2.16429					COF: 3.30081		
8	325	2	846.64	8.49	18	151	2	879.84	12.0
		COF: 2.96664					COF: 3.31366		
9	160	2	829.17	8.49	19	173	3	839.0	8.07
		COF: 2.96913					COF: 3.9263		
10	382	2	851.76	9.02	20	75	3	841.28	9.44
		COF: 3.01986					COF: 4.06275		

Table 5 shows how much close the voting of the instance #407 (rank 1) to its neighbors. Similarly, it shows how much close the voting of instance #375 (rank 2) to its neighbors. It is also clear from the table that the votes of instance #176 are becoming more different from the votes of its democrats neighbors. This made instance #176 in the rank 11.

TABLE V
THE 7NN OF RANK 1, 2, AND 11 CLASS OUTLIERS OF HOUSE-VOTE-84 DATASET

Inst.#		*KDist*
	Rank 1	
	The 7 Nearest Neighbors of the Instance #407	
407	n,n,n,y,y,y,n,n,n,n,y,y,y,n,n,democrat	0.0
306	n,n,n,y,y,y,n,n,n,n,n,y,y,y,n,n,republican	1.0
83	n,n,n,y,y,y,n,n,n,n,n,y,y,y,n,n,republican	1.0
87	n,n,n,y,y,y,n,n,n,n,n,y,y,y,n,n,republican	1.0
303	n,n,n,y,y,y,n,n,n,n,n,y,y,y,n,n,republican	1.0
119	n,n,n,y,y,y,n,n,n,n,n,y,y,y,n,n,republican	1.0
339	y,n,n,y,y,y,n,n,n,n,y,y,y,y,n,n,republican	1.0
	Rank 2	
	The 7 Nearest Neighbors of the Instance #375	
375	n,y,n,y,y,y,n,n,n,n,y,y,n,y,n,n,democrat	0.0
324	n,y,n,y,y,y,n,n,n,n,y,y,y,y,n,n,republican	1.0
154	n,y,n,y,y,y,n,n,n,n,n,y,y,y,n,n,republican	1.41
55	n,y,n,y,y,y,n,n,n,y,y,y,y,y,n,n,republican	1.41
30	n,y,n,y,y,y,n,n,n,n,n,y,y,y,n,n,republican	1.41
35	n,y,n,y,y,y,n,n,n,n,n,y,y,y,n,n,republican	1.41
61	n,y,n,y,y,y,n,n,n,n,n,y,y,y,n,n,republican	1.41
	Rank 11	
	The 7 Nearest Neighbors of the Instance #176	
176	n,n,y,y,n,n,y,y,y,y,n,n,n,y,y,y,republican	0.0
280	n,n,y,n,n,n,y,y,y,y,n,n,n,y,n,y,democrat	1.41
110	n,n,y,n,n,n,y,y,y,y,n,n,n,n,y,y,democrat	1.41
200	n,n,y,n,n,n,y,y,y,n,n,n,n,y,y,y,democrat	1.41
255	y,n,y,n,n,n,y,y,y,y,n,n,n,y,y,y,democrat	1.41
355	y,n,y,y,n,n,n,y,y,y,n,n,n,y,y,y,republican	1.41
338	y,n,y,n,n,n,y,y,y,y,n,n,n,y,y,y,democrat	1.41

B. Experiment II (vehicle dataset)

The dataset of vehicle [5] contains 846 instances belonging to four classes, i.e. opel, saab, bus and van, described by 19 attributes (including the class label attribute) among which 18 attributes are continuous. The class distribution is 212 opel, 217 saab, 218 bus and 199 van (25.06% opel, 25.65% saab, 25.77% bus, 23.52% van). There are no missing values in the dataset. This experiment is very interesting because all the attributes of the dataset are continues. Furthermore, the dataset includes multiple class labels which are almost equally distributed. The following inputs are provided to the implemented algorithm:

$K = 9$, *Top N COF* = 10, $\alpha = 100$, $\beta = 0.1$, *Remove Instance with Missing Values:* false, *Distance type: Euclidean* Distance, *Replace Missing Values:* false.

TABLE VI
THE TOP 10 CLASS OUTLIERS OF VEHICLE DATASET

#	*Inst.#*	*PCL*	*Dev*	*KDist*	*#*	*Inst.#*	*PCL*	*Dev*	*KDist*
1	349	1	289.25	2.61	6	32	1	257.48	2.48
		COF: 1.60661					COF: 1.63681		
2	216	1	233.31	1.93	7	806	1	216.87	2.05
		COF: 1.6217					COF: 1.66612		
3	599	1	263.36	2.43	8	645	1	218.67	2.25
		COF: 1.62245					COF: 1.68263		
4	422	1	293.19	2.82	9	113	1	361.99	4.15
		COF: 1.62287					COF: 1.69109		
5	163	1	243.16	2.16	10	451	1	230.82	2.72
		COF: 1.6273					COF: 1.7051		

Table 6 shows the top 10 *class outliers* whereas table 7 shows the distance of rank 1, 3, and 5 *class outlier* instances from its *K* nearest neighbors (9 *NN*). Referring to table 6 and 7, consider the instance #349 (rank 1, class: opel) which is chosen at the top despite that there is more than only one class labels of its surrounding. It is to be noticed that the *PCL* of its surrounding is 2/9 (saab) and 6/9 (van). Comparing this case with instance #599 (rank 3), where there is only one class of its surrounding, (i.e., the *PCL* of its surrounding is 8/9 (van)). The justification can be extracted from table 6, where the *Deviation* is 289.25 and 263.36 for the instances #349 and #599 respectively.

We performed more investigations about the instance #349's neighbors, especially the surrounding instances that have second minority *PCL*, which is 2/9 (saab), these instances are 636, and 148. We found that both *PCL(636, 9)* and *PCL(128, 9)* are 2/9, which means they are also considered as *class outliers* but in an order, which is beyond 10.

Comparing instances #349 (rank 1) and #163 (rank 5), they are almost similar (the *PCL* of their surrounding is 2/9 and 6/9), but the *Deviation* of the instance #163 is 243.16, which is less than the *Deviation* of the instance #349, and the difference of *KDist* for both the instances is not much. This does not allow the *KDist* to make a direct impact in the ranking. The explanation described above reflects the importance of *Deviation* factor.

TABLE VII
THE 9NN OF RANK 1, 3, AND 5 CLASS OUTLIERS OF VEHICLE DATASET

Inst.#		*KDist*
	Rank 1	
	The 9 Nearest Neighbors of the Instance #349	
349	89,40,69,147,58,6,132,50,18,137,155,260,151,61,16,6,203,209,opel	0.00
460	90,41,62,147,60,6,128,52,18,141,149,246,157,61,13,4,201,208,van	0.22

Inst.#		*KDist*
469	92,40,62,144,59,8,127,52,17,139,149,241,150,62,13,1,204,210,van	0.25
703	93,43,78,162,64,8,137,48,18,145,156,281,159,63,17,12,203,210,van	0.30
629	90,42,63,144,59,7,131,50,18,142,154,259,162,65,15,3,197,204,van	0.32
636	96,41,69,153,56,7,141,47,18,141,162,297,169,61,11,8,202,209,saab	0.33
403	96,39,77,160,62,8,140,47,18,150,161,294,124,62,15,3,201,208,van	0.36
148	90,43,72,172,59,8,154,42,19,144,174,360,158,61,15,9,203,209,saab	0.39
330	98,44,78,160,63,8,142,47,18,148,160,300,171,63,19,2,201,207,van	0.42
	Rank 3	
	The 9 Nearest Neighbors of the Instance #599	
599	93,39,63,146,58,7,128,52,18,134,149,246,158,63,9,7,198,204,saab	0.00
204	89,40,58,137,58,7,122,54,17,140,146,225,150,63,7,4,199,206,van	0.24
268	86,39,60,140,60,7,119,55,17,134,140,212,141,61,7,8,200,207,van	0.29
460	90,41,62,147,60,6,128,52,18,141,149,246,157,61,13,4,201,208,van	0.30
607	86,39,62,129,59,6,116,57,17,135,137,203,145,64,7,9,199,204,van	0.30
754	91,41,64,148,61,8,129,51,18,142,161,249,153,68,6,12,194,201,van	0.31
262	89,40,60,131,56,6,118,56,17,137,143,209,153,65,10,8,193,199,van	0.32
537	86,40,66,139,59,7,122,54,17,139,145,225,143,63,7,11,202,208,van	0.33
55	94,36,66,151,61,8,133,50,18,135,154,265,119,62,9,3,201,208,van	0.35
	Rank 5	
	The 9 Nearest Neighbors of the Instance #163	
163	85,40,72,139,59,5,132,50,18,135,159,260,150,68,3,9,191,195,saab	0.00
483	86,38,76,143,59,8,142,47,18,131,167,301,138,71,5,10,189,196,van	0.23
316	91,41,66,131,56,9,126,53,18,144,159,237,155,72,3,10,191,194,van	0.26
340	89,40,72,155,63,7,146,45,19,135,175,321,145,72,4,10,192,196,bus	0.27
286	83,41,70,155,65,7,144,46,19,141,168,309,147,71,4,12,188,195,bus	0.27
211	86,37,69,150,63,8,138,48,18,134,163,284,124,71,1,6,189,195,van	0.27
46	91,43,70,133,55,8,130,51,18,146,159,253,156,70,1,8,190,194,van	0.28
774	94,37,72,146,60,9,133,50,18,135,161,262,128,69,2,7,192,195,van	0.29
514	89,38,74,138,59,7,136,49,18,133,167,278,128,72,7,7,189,193,van	0.29

C. Experiment III (hepatitis dataset)

The dataset of hepatitis [5] contains 155 instances belonging to two classes, i.e. positive or negative for hepatitis, described by 20 attributes (including the class label attribute) among which 6 attributes are continuous and the remaining 13 attributes are categorical. The class distribution is 32 DIE, 123 LIVE (20.67% DIE, 79.35% LIVE).

Table 8 shows the top 10 *class outliers*. The following inputs are provided to the implemented algorithm:

K = 7, *Top N COF* = 10, *α* = 10, *β* = 0.1, *Remove Instance with Missing Values:* false, *Distance type: Euclidean* Distance, *Replace Missing Values:* true.

TABLE VIII
THE TOP 10 CLASS OUTLIERS OF HEPATITIS DATASET

#	*Inst.#*	*PCL*	*Dev*	*KDist*	#	*Inst.#*	*PCL*	*Dev*	*KDist*
1	31	1	65.89	5.11	6	120	2	69.15	7.47
		COF: 1.66273					COF: 2.89204		
2	35	1	69.02	6.71	7	71	2	70.37	7.76
		COF: 1.81633					COF: 2.91841		
3	134	1	75.53	7.72	8	30	2	75.59	9.06
		COF: 1.90424					COF: 3.03814		
4	98	1	72.95	8.3	9	76	2	76.99	9.91
		COF: 1.96694					COF: 3.12085		
5	128	2	269.39	7.02	10	126	2	296.85	7.21
		COF: 2.73911					COF: 3.75509		

D. Experiment IV (heart-statlog)

The dataset of heart-statlog [5] contains 270 instances belonging to two classes, i.e. absent or present for heart disease, described by 14 attributes (including the class label attribute) among which 13 attributes are continuous. The class distribution is 150 absent, 120 present (55.56% absent, 44.44% present). There are no missing values in the dataset. Table 9 shows the top 10 *class outliers*. The following inputs are provided to the implemented algorithm:

K = 7, *Top N COF* = 10, *α* = 100, *β* = 0.1, *Remove Instance with Missing Values:* false, *Distance type: Euclidean* Distance, *Replace Missing Values:* false.

TABLE IX
THE TOP 10 CLASS OUTLIERS OF HEART-STATLOG DATASET

#	*Inst.#*	*PCL*	*Dev*	*KDist*	#	*Inst.#*	*PCL*	*Dev*	*KDist*
1	69	1	206.0	1.61	6	169	1	234.94	5.01
		COF: 1.64619					COF: 1.92677		
2	11	1	276.9	3.2	7	67	1	247.44	5.32
		COF: 1.6814					COF: 1.93567		
3	207	1	268.89	3.66	8	175	1	250.37	7.69
		1.73806					COF: 2.16821		
4	258	1	205.36	2.85	9	3	2	277.4	2.75
		COF: 1.77229					COF: 2.63559		
5	177	1	202.1	4.2	10	91	2	220.28	2.1
		COF: 1.91521					COF: 2.66381		

E. Experiment V (credits approval)

The dataset of credit approval (credit-a) [5] contains 690 instances belonging to two classes, i.e. "+" or "-" for credit approval, described by 16 attributes (including the class label attribute) among which 6 attributes are continuous and the remaining 10 attributes are categorical. The class distribution is 307 "+", 383 "-" (44.49% "+", 55.51% "-"). All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data.

This dataset is interesting because there is a good mix of attributes: continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also few missing values. Table 10 shows the top 10 *class outliers*. The following inputs are provided to the implemented algorithm:

K = *7*, *Top N COF* = 100, *α* = 10, *β* = 0.1, *Remove Instance with Missing Values:* false, *Distance type: Euclidean* Distance, *Replace Missing Values:* false.

TABLE X
THE TOP 10 CLASS OUTLIERS OF CREDIT-A DATASET

#	*Inst.#*	*PCL*	*Dev*	*KDist*	#	*Inst.#*	*PCL*	*Dev*	*KDist*
1	115	1	833.32	.85	6	48	1	669.1	2.77
		COF: 1.2047					COF: 1.42608		
2	523	1	837.58	1.12	7	348	1	820.44	3.73
		COF: 1.23097					COF: 1.49474		
3	110	1	806.66	1.65	8	546	1	877.5	4.42
		COF: 1.28872					COF: 1.55615		
4	99	1	841.87	2.26	9	116	1	865.02	4.58
		COF: 1.34454					COF: 1.57336		
5	320	1	841.87	2.26	10	13	1	636.3	4.55
		COF: 1.34454					COF: 1.61253		

V. COMPARISON STUDY

In this section, we perform a comparative study with He's method [13, 14]. In the following, we shall try to abstract the main difference between *SOF* formula and our proposed *COF* formula.

Figure 13 illustrates the difference between the ranking criteria of our proposed approach and semantic outlier (*COF* vs. *SOF*). Suppose the size of both the clusters A and B is 100, and the probability of the class *x* with respect to the cluster is 3/100 for both the cases A and B. In *SOF* approach x_1 in both the cases A and B has the same rank, but using *COF*, the rank is different because *PCL* of x_1 is 3/7 for the case A, and 1/7 for the case B (assuming *K* = 7).

We shall compare the results obtained by using *SOF* and *COF* applied on house-vote-84 dataset. The house-vote-84 dataset is being chosen for comparison because it is the only available dataset tried by *SOF* approach. The experimental results obtained by using *SOF* on the house-vote-84 dataset are shown in table 11. Comparing our results obtained in table 4 using our approach with table 11, we notice that the instance #176 in house-vote-84 dataset is the top (rank 1) outlier using *SOF* (table 11) whereas using the *COF* the rank for the same instance is 11 (table 4). It is to be noticed that the *PCL* of the instance is 2/7 which indicate that there is another instance of the same class within the seven nearest neighbors. This case is considered a similar case to that which described in Figure 14.

Instance #407 is ranked first using *COF* while it has the rank 9 using *SOF*. From our observation, instance #407 is alone of its class type among seven nearest neighbors. Moreover, its *Deviation* is the greatest, which implies sort of uniqueness of the instance (object) behavior. The *K-Distance* of the instance is very small (high density of other class type). In *SOF* ranking 9 of the instance #407 indicates the disability of recognizing such important cases.

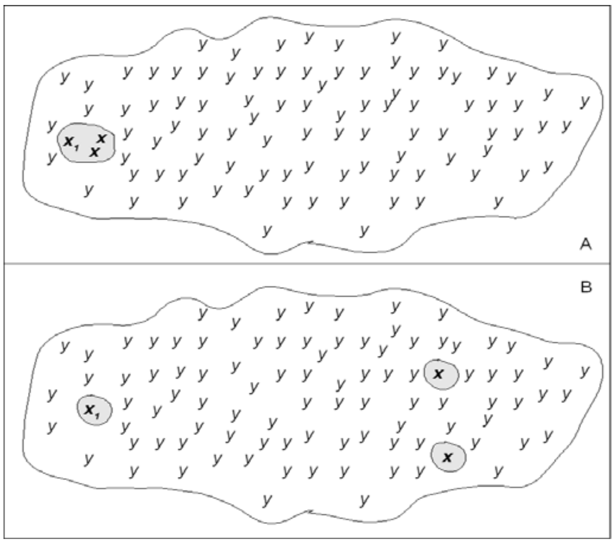

Fig. 13: *SOF* vs. *COF*

TABLE XI
THE TOP 20 SEMANTIC OUTLIERS FOR HOUSE-VOTE-84 DATASET

#	*Inst. #*	*SOF*	#	*Inst. #*	*SOF*
1	176	0.3036	11	375	1.4520
2	71	0.3394	12	151	1.4927
3	355	0.3645	13	372	1.4950
4	267	0.3659	14	388	1.6365
5	183	0.8726	15	2	1.6489
6	97	0.9892	16	382	1.6727
7	88	1.0724	17	215	1.7010
8	402	1.1690	18	164	1.7168
9	407	1.3309	19	6	1.7236
10	248	1.3487	20	325	1.7259

A sample of instances that are listed in our approach's rank (table 4) and not listed in *SOF's* rank (table 11) are instances #161 (*PCL* = 1/7, *Deviation* = 819.35, *KDist* = 8.49, *COF* = 1.97) and #77 (*PCL* = 1/7, *Deviation* = 799.64, *KDist* = 10.39, *COF* = 2.16). Instances #161, and #77 which have rank 4 and 7 respectively using *COF*. But as we stated above, instance #161 and #77 are not listed in top 20 using *SOF* which means *SOF* could not detect such cases even in a later order.

Instance #183 (*PCL* = 5/7, *Deviation* = 1009.86, *KDist* = 14.56, *COF* = 6.55) is ranked 5 using *SOF* and not listed in top 20 using *COF*. The *PCL* is 5/7 which means that there are similar cases to this case surrounding it. The *Deviation* is considered very high and this is expected because most of his votes about the issues are "noVote". Instance #97 (*PCL* = 5/7, *Deviation* = 766.9, *KDist* = 11.15, *COF* = 6.24) is ranked 6 using *SOF* and not list in top 20 using *COF*. The *PCL* is 5/7 which means that there are similar cases to this case surrounding it.

We believe that ranking instances using *COF* which use *KNN* approach, gives more reasonable results.

VI. CONCLUSION

We have presented a novel approach for *Class Outliers Mining* based on the *K* nearest neighbors using distance-based similarity function to determine the nearest neighbors. We introduced a motivation about *Class Outliers* and their significance as exceptional cases. We proposed a novel definition for *Class Outlier* and a ranking score that is *Class Outlier Factor* (*COF*) to measure the degree of being a *Class Outlier* for an object. The main key factors of computing *COF* are the probability of the instance's class among its neighbors's classes, the deviation of the instance from the instances of the same class, and the distance between the instance and its k neighbors.

Beyond the problem definition and motivation, we proposed new algorithm for mining and detecting *Class Outliers*. An implementation has been developed with the help of *Weka* framework [28] in certain functions, other functions have been implemented using Java. The algorithm implementation software has been tested with various real domain datasets (medical, business, and other domains), and for different dataset types (continues, nominal with small numbers of values, nominal with larger numbers of values, and mixed). The experimental results were very interesting and reasonable. Furthermore, a comparison study has been performed with related methods.

In our proposed algorithm, we assume that all the features (attributes) are equally important for computing *COF,* it is well known that some features may be more important than others for class label. Referring to Figure 4 and tables 2, features {4, 3, 5, 12, 8} are more important than the other features. In the future work, we shall develop a weighted distance similarity function, where feature weight determination might be based on the information gain. Furthermore, we shall propose a *Class Outlier Detection Model.* In addition, the output of this work could be very useful to find out a scheme to induce Censored Productions Rules (*CPRs*) [23] from large datasets.

REFERENCES

[1] Angiulli, F., Pizzuti, C.: *Fast Outlier detection in high dimensional spaces*, In Proc. of the Sixth European Conference on the Principles of Data Mining and Knowledge Discovery, pp. 15-26, 2002.

[2] Barbarà, D., Chen, P.: *Using the fractal dimension to cluster datasets*, In: Proc. KDD, pp. 260–264, 2000.

[3] Barnett, V., Lewis, T.: *Outliers in Statistical Data*, John Wiley, 1994.

[4] Bay, S. D., and Schwabacher, M.: *Mining Distance-Based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule*, Proc. of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003.

[5] Blake C., Keogh E., Merz C. J.: *UCI Repository of Machine Learning Databases*, http://www.ics.uci.edu/~mlearn/MLRepository.htm, 1998.

[6] Bolton, R. J., Hand, D. J.: *Statistical fraud detection: A review (with discussion)*, Statistical Science, 17(3): pp. 235-255, 2002.

[7] Breunig, M., Kriegel, H., Ng, R., Sander, J.: *LOF: Identifying density-based local outliers*, In: Proc. SIGMOD Conf, pp. 93–104, 2000.

[8] Eskin E., Arnold A., Prerau M., Portnoy L., Stolfo S.: *A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data*, In Data Mining for Security Applications, 2002.

[9] Ester M., Kriegel H.-P., Sander J., Xu X.: *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*, Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, OR. pp. 226-231, 1996.

[10] Han, J., Kamber, M.*: Data Mining: Concepts and Techniques*, San Francisco, Morgan Kaufmann, 2001.

[11] Hawkins, D.: *Identification of Outliers*, Chapman and Hall, 1980.

[12] Hawkins, S., He, H. X., Williams, G. J., Baxter, R. A.: *Outlier detection using replicator neural networks*, In Proc. of the Fifth Int. Conf. and Data Warehousing and Knowledge Discovery (DaWaK02), 2002.

[13] He, Z., Deng, S., Xu., X.: *Outlier detection integrating semantic knowledge*, In: Proc. of WAIM'02, pp. 126-131, 2002.

[14] He, Z., Xu, X., Huang, J., Deng, S.: *Mining Class Outliers: Concepts, Algorithms and Applications in CRM*, Expert Systems with Applications (ESWA'04), 27(4): pp. 681-697, 2004.

[15] Jain, A., Murty, M., Flynn, P.: *Data clustering: A review*, ACM Comp, Surveys 31, 264–323, 1999.

[16] Johnson, T., Kwok, I., Ng, R.: *Fast computation of 2-dimensional depth contours*, In: Proc. KDD. pp. 224–228, 1998.

[17] Knorr E. M., Ng. R. T.: *Finding intensional knowledge of distance-based outliers*, In Proc. of the 25th VLDB Conference, 1999.

[18] Knorr, E., Ng, R., Tucakov, V.: *Distance-based outliers: Algorithms and applications*, VLDB Journal 8, pp. 237–253, 2000.

[19] Knorr, E., Ng, R.: *A unified notion of outliers: Properties and computation*, In: Proc. KDD. pp. 219–222, 1997.

[20] Knorr, E., Ng, R.: *Finding intentional knowledge of distance-based outliers*, In: Proc. VLDB. pp. 211–222, 1999.

[21] Knorr, E.M., Ng, R.: *Algorithms for mining distance-based outliers in large datasets*, In: Proc. VLDB pp. 392–403, 1998.

[22] Lane, T., Brodley, C. E.: *Temporal sequence learning and data reduction for anomaly detection*, ACM Transactions on Information and System Security, 2(3): pp. 295-331, 1999.

[23] Michalski, R. S., Winston, P. H.: *Variable Precision Logic*, Artificial Intelligence Journal 29, Elsevier Science Publishers B.V. (North-Holland), pp. 121-146,1986.

[24] Papadimitriou, S., Faloutsos C.: *Cross-outlier detection*, In: Proc. of SSTD'03, pp. 199-213, 2003.

[25] Ramaswamy, S., Rastogi, R., Shim, K.: *Efficient algorithms for mining outliers from large data sets*, In Proc. of the ACM SIGMOD Conference, pp. 427-438, 2000.

[26] Rousseeuw, P., Leroy, A.: *Robust Regression and Outlier Detection*, John Wiley and Sons, 1987.

[27] Rulequest Research, Gritbot, http://www.rulequest.com

[28] Witten, I. H., Frank, E.: Data Mining: *Practical Machine Learning Tools and Techniques*, (Second Edition), San Francisco, Morgan Kaufmann, 2005.

Nabil M. Hewahi is a professor of computer science, Islamic university of Gaza, Palestine. He obtained his B.Sc. in computer science from Al-Fateh University, Libya in 1986 and M.Tech. degree in computer science and engineering from Indian institute of technology, Bombay, India. In 1994 he obtained his PhD in computer science from Jawaherlal Nehru University, New Delhi, India. Dr. Hewahi worked in academics and administration. He is currently the dean of the faculty of information technology at the Islamic university of Gaza, Palestine. Dr. Hewahi has published several papers in well known conferences and journals. His research interests include genetic algorithms, neural networks, neuroevolution, soft computing, knowledge representation, and machine learning.

Motaz K. Saad is a teaching/research assistant at the department of computer science, Islamic university of Gaza- Palestine. Mr. Saad obtained his BSc in computer science from the Islamic university of Gaza in 2006. He is a Microsoft Certified Professional (MCP), and highly trained in various fields such as networking and software & algorithm design. His research main interests are data mining, artificial intelligence, and machine learning.