Class-based outlier detection: staying zombies or awaiting for resurrection?

Leona Nezvalová, Luboš Popelínský, Luis Torgo, and Karel Vaculík

KD Lab, FI MU Brno and F.Sci. U.Porto
xnezva36@mail.muni.cz, popel@fi.muni.cz, ltorgo@dcc.fc.up.pt,
xvaculi4@fi.muni.cz

Abstract. This paper addresses the task of finding outliers within each class in the context of supervised classification problems. Class-based outliers are cases that deviate too much with respect to the cases of the same class. We introduce a novel method for outlier detection in labelled data based on Random Forests and compare it with existing methods both on artificial and real-world data. We show that it is competitive with the existing methods and sometimes gives more intuitive results. We also provide an overview for outlier detection in labelled data. The main contribution are two methods for class-based outlier description and interpretation.

1 Introduction

Outlier detection [2] is an area of data analysis that aims at finding anomalies in data. Data cleansing to improve statistical model fraud detection or computer network intrusion detection are examples of some successful application areas of outlier analysis. Main stream of outlier detection defines, for a given statistical distribution, an outlier (or a series of outliers in the case of contextual outliers) as a case that differs from normal cases. This is typically taken as an unsupervised task in the sense that there are no labels indicating to the models which are the normal and the outlier cases. Nevertheless, there are also approaches that assume the existence of a supervised training set with examples of outlier cases and normal cases where the goal is to obtain a model that is able to accurately discriminate these two situations. Moreover, semi-supervised approaches have also been used in these application areas, where only a part of the available data is labelled.

In this paper we address another type of use of the concept of outliers. The context of the problem we are tackling is that of standard supervised classification tasks. Cases are labelled into a set of pre-defined classes in these problems and the goal is to learn a classification model from a provided training set. Outlier detection in labelled data, as a specific task, was initiated in [9, 16] and further elaborated in [8]. Class-based outliers are those cases that look anomalous when the class labels are taken into account, but they do not have to be anomalous when the class labels are ignored. In [8] two class outlier detection

methods have been introduced as adaptations of methods for classical outlier detection setting, one based on frequent pattern mining, the other on clustering, and their use for Custom Relation Management was demonstrated. A distance and density-based approach has been published in [10] and its slightly improved version is now available in RapidMiner. Usability of these methods in Custom Relation Management and also in educational data was demonstrated [8, 18].

In [3] a novel unsupervised way of detecting outliers for two-class problems by Inductive Logic Programming is presented. In the following text we will call it CB-ILP. The essential idea is that the outliers somehow disrupt the model of the data. The detection is done by creating a model, then for each possible outlier (or a set of outliers) excluding this outlier(s), learning a new model and comparing it with the original model. The same is done for a dual problem where positive and negative examples have been swapped. If coverages of two learned rule sets (with and without an example) differ more than a threshold, the example is an outlier. This approach then allows, by comparison of coverages of those four learned models, us to characterize the anomalies more finely, and to divide outliers into three groups according to the way they disrupt models learned with the whole data set - Irregularities, Anomalies and Outliers. The main drawback is its computational complexity.

Class-based outlier detection can be seen similar to label noise detection in classification tasks [7], more precisely to its sub-part of noise elimination. The first work that exploited class-based outliers, although not explicitly named, was C45Robust [11], where detected misclassified cases were removed before learning a new model. A similar idea has been recently elaborated for SVM [17]. The most important difference from our approach lies in a different measure of interestingness (rank). For noise elimination it seems to correspond only to the influence of a potential outlier to learning the correct hypothesis. In class-based outlier detection, the detection of an outlier is the goal and its interestingness may not depend only on classification accuracy but also on its novelty. Class-based outlier detection is also close to works on Exceptional model mining [13].

In this paper, we present RF–OEX, a new approach to class-based outlier detection based on Random Forests (RF) implemented in Weka [6]. It consists of two parts, an outlier detection module and an outlier interpretation module. We compare outliers obtained with RF–OEX with results of eCODB [10] and CB–LOF, an adaptation of LOF to labelled data. We show that RF–OEX is comparable with them, both on artificial data and real-world data, or gives even more intuitive results. Two new methods for class-based outlier interpretation, one based on tree reduction, the other on finding frequent branches in those trees, have been implemented. We describe these methods and compare the results with CB-ILP, an ILP approach to outlier detection [3].

In the following section we give an overview of the class-based outlier detection problem. In Section 3 we describe class outlier detection with RF–OEX, experimental evaluation of RF–OEX and comparison with the other methods. Two novel methods for outlier interpretation are described in Section 4. There we also bring a review of existing methods for class-based outlier interpretation

and compare the results of RF–OEX with their interpretation. We conclude with discussion of results and directions for future work. Supplementary information can be found at http://www.fi.muni.cz/~popel/685269.

2 Class-based outlier detection

Class-based outliers are cases that are *different* from the rest of the members of their own class. There are two main types of ways of being different from the members of the same class:

1. Cases that are too far (in terms of distance) from the bulk of cases of the same class.
2. Cases that, although not too far from the same class, are nearer to cases of another class.

Fig. 1 illustrates both situations. (a) shows the first situation, where the case in red is too far from the members of the same class (x), while (b) illustrates the second situation, where the red case is nearer to members of the other class (y), even though it is not too far from the members of the same class (x), and thus regarded by us as a class-based outlier.

Fig. 1. The first (a) and the second (b) type of class-based outliers.

While the case (a) could be easily detected by standard distance-based outlier detection algorithms, the case (b) would not be captured by these methods as they are class-agnostic. This means that detecting this second type of class-based outliers is the key issue / innovation on this research line.

The concept of class outliers – outliers in labelled data – was introduced in [8] as a generalisation of two concepts: *semantic outliers*, i.e. data points that differ from other members of the same class while looking normal (similar) to data points in another class [9], and *cross-outliers* that deviate from data points in another class [16]. In [8] the authors define the problem of **Class Outlier Detection** as finding observations (data points) with class labels that arouse suspicions when those class labels are taken into account. Let

- C_i be a set of observations with the same class label cl_i,
- $DB = \{C_1, C_2, ..., C_m\}, C_i \cap C_j = \emptyset$ for $i \neq j$, be a data set,
- $Sp(DB)$ be the set of all unions of subsets of DB,

- T be an element of DB (i.e. $T = C_j$ for $j \in \{1, 2, ..., m\}$),
- $p \in T$ where p is an observation and T a target set.

The ***Class Outlier Detection*** problem is to find for two classes C_i, C_j, $i \neq j$, those observations $p \in C_i$ that differ from members of C_i and C_j. Let $COF(p, C_i)$ be the class outlier factor for $p \in T$. If $C_i = T$, i.e. the problem is to find those observations $p \in T$ that differ from other members of the same class T, then we call it ***Local Class Outlier Detection*** and use $LCOF(p)$ as the local class outlier factor. If $C_i \neq T$, i.e. the problem is to find those observations $p \in T$ that differ from members of the other class C_i, then we call it ***Reference Class Outlier Detection*** and use $RCOF(p, C_i)$ as the reference class outlier factor. If $DB = T$ then the class outlier detection problem is reduced to the common outlier detection problem.

For example, assume a two-class problem, where C_1 are positive and C_2 negative examples, $p \in C_1$, then p may be a local outlier, i.e. a reference class outlier w.r.t class C_1 with class outlier factor $LCOF(p) = RCOF(p, C_1)$ (or a semantic outlier in terms of [9]). It may be a reference class outlier w.r.t class C_2 with class outlier factor $RCOF(p, C_2)$ (or a cross-outlier by [16]).

The main idea presented in [8] lies in the computation of class-based outlier factor as an aggregation of the local factor and reference factors.

3 RF–OEX: Outlier detection

3.1 Method

Random Forests [4] is an ensemble method that combines bagging (each tree is constructed by a different bootstrap sample from the original data) with the idea of random selection of features. More specifically, each split of a tree is chosen from a random subset of the data set features. Only these features are then used for selecting the best split at each node and this random process is repeated for all splits and all trees.

Random Forests can be used as an outlier detection method[1] in the following way. After each tree is built, all of the data are run down the tree, and proximity values are computed for each pair of cases. If two cases occupy the same terminal node, their proximity is increased by one. At the end of the run, the proximity values are normalized by dividing by the number of trees and the average proximity is computed for each instance.

The main idea of RF–OEX lies in a different way of exploitation of the proximity matrix. The main difference then lies in the fact how RF–OEX exploits the information about the class label. The outlier factor for an instance p is computed as a sum of three different measures of proximity or outlierness – proximity to the members of the same class OF_1, misclassification measure (proximity to the members of other classes) OF_2 and ambiguity measure OF_3. A similar idea, but only for the first two addends, has been elaborated in [8]. In the following,

[1] https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#outliers

p stands for an element for which we compute the outlier factor. The value of the outlier factor is given by:

$$OF(p) = OF_1(p)_{same-class} + OF_2(p)_{misclassification} + OF_3(p)_{ambiguity}$$

$\boldsymbol{OF_1(p)_{same-class}}$. In this case, only proximities to points from the same class are taken into account. Proximity of point p from class C_p is computed as an aggregation of proximities to all points from the same class. Four aggregation functions have been implemented: *sum, sum of squared proximity values, product, and cube root of sum of cubic values.* For simplicity, we will use *sum* function in the resulting formula. In principle, the higher the proximity is, the lower its outlierness is, so we use the inverse value of the proximity: $OF_1(p) = \frac{1}{\sum_{cl(p)=cl(q)} Prox(p,q)}$. where $cl(p)$ is the class label of element p. Finally, we normalize the result because of different sizes of different classes.

$\boldsymbol{OF_2(p)_{misclassification}}$. We already stated that the similarity with members of a different class should increase the class outlier factor of p. We define $Top_{|C_p|}(p)$ as $|C_p|$ poins that are closest to point p. Then we compute how many of those points are labelled by different class than point p belongs to. To be comparable with OF_1 and OF_3, the value is multiplied by constant c, which is computed as the maximum from all values OF_1 divided by 4: $c = \frac{max_{q \in DB} OF_1(q)}{4}$. The resulting formula is then defined as $OF_2(p) = c \cdot \frac{|\{q \mid cl(q) \neq cl(p) \ \&\ q \in Top_{|C_p|}(p)\}|}{|C_p|}$.

$\boldsymbol{OF_3(p)_{ambiguity}}$. To increase the importance of outliers that are far from all points, we add the third addend OF_3. We compare the sum of proximites of points $Top_{|C_p|}(p)$ to point p with the ideal situation when proximity to all examples is 1 and the sum is equal to $|C_p|$. At the end, we multiply it with the same constant as in the case of OF_2: $OF_3(p) = c \cdot \frac{|C_p| - \sum_{q \in Top_{|C_p|}(p)} Prox(p,q)}{|C_p|}$.

3.2 Parameter settings

The RF–OEX method has a few parameters but in most cases the user does not need to change the default values of these parameters. At most the user may need to try a few alternatives for the parameter that controls how many random features are used for split selection at each tree node. Below we describe the values used for the parameters of our method. More information on parameter settings and work with RF–OEX can be found on the supplementary web page.

Number of Trees was set to 1000. We also checked smaller values, between 100 and 1000, and for a lot of situations 100 was sufficient. *Number of Random Features* for each split selection step was set to half of the number of input attributes. *Minimum cases per node* is equivalent to *-m* parameter of C45. It does not allow to grow the tree when few examples reached the node. We set this parameter to 10 for real-world data sets and 0 for artificial data because they contained very few instances. *Maximum depth of Trees* was set to 0, i.e. depth of trees was not limited. *Attribute distribution of multiset for Random tree* was set to *Normal*, so each tree starts with the same original set of attributes. The

information gain of attributes is then taken into account before building each node of tree. *Variant of summing point's proximities* denotes the method of summing sample proximities and was set to *Addition squared values*. *Normalize according to* affects the normalization of outlier factor within the bounds of experiment. The *Average* option was chosen for this parameter. We checked both *Count with mistaken class penalty* and *Count with ambiguous classification penalty* parameters to consider similarity of given instance with the rest of samples ($OF_2(p)_{misclassification}$ and $OF_3(p)_{ambiguity}$) when calculating the outlier factor. The *Use data bootstrapping* parameter was checked to generate trees of different quality.

3.3 Data sets used for experiments

For experiments we used the following data sets.

Artificial data. We created several artificial data sets – two-class problems, with two numeric attributes – to check the behaviour of the systems on controlled situations. An example of such a dataset with two classes is in Fig. 2, all the data sets can be found at `http://www.fi.muni.cz/~popel/685269/Results/OutlierDetection/top_5_artificial.pdf`.

Votes. The Votes (or House Votes 84 – Republicans vs. Democrats) data set contains 16 key congress votes for each U.S. House of Representatives Congressman, 267 democrats and 168 republicans and was also used in [8, 10]. It has been observed that several congressmen have opinions almost exactly opposite to what is common in their political party (meaning they vote similarly to their political opponents).

Student solutions data. The data [18] (a total of 873 student solutions) was obtained in a bachelor course on Introduction to Logic at FI MU. It contains the resolution tree together with dynamics of the solutions, i.e. all the actions performed together with temporal information. Here we use only the resolution trees. First all subgraphs that correspond to an application of the resolution rule were found and generalized. Then each resolution tree has been transformed to 0/1 matrix where those graphs served as boolean attributes (1=the subgraph appeared in the tree, 0=otherwise). Among these 873 different students' solutions of resolution proofs in propositional calculus, 101 of them were classified as incorrect and 772 as correct [18].

As calculation of proximity matrix is quadratic to the number of instances, the method becomes time-consuming for big data sets. Therefore, we have restricted our selection of data sets to those having less than 1000 instances. For example, runtime of Votes data set, which has 423 instances, is less than one minute. Running the student solutions data set, which has 873 instances, takes 10 minutes. Optimization of our method to be applicable to bigger data sets is part of future extensions.

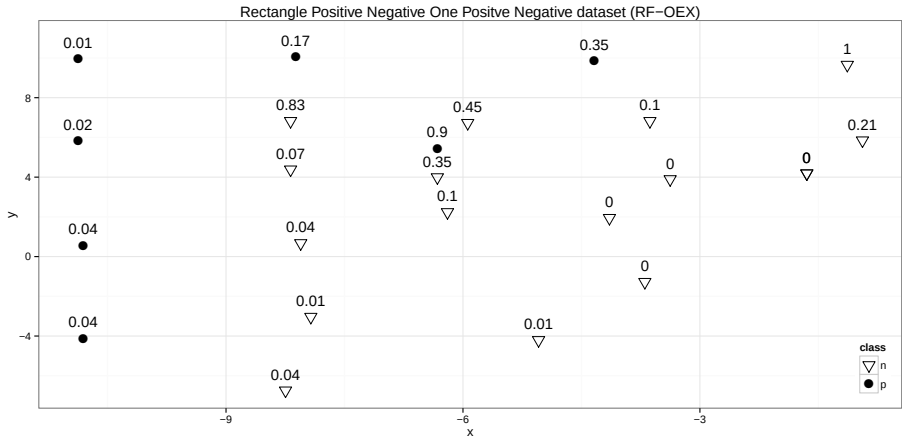

Fig. 2. Results of RF–OEX on an artificial data set; outlier factors are above the datapoints and they are normalized to [0,1] interval, where 1 means the most outlying.

3.4 Experiments and results

We compared results of RF–OEX with eCODB [10] and a variant of LOF that follows the model described in Section 2. eCODB [10] combines distance-based and density-based approach w.r.t class attribute. The Class Outlier Factor $COF(T)$ for an instance T and parameter K (K nearest neighbors of the instance T) is computed as

$$COF(T,K) = PCL(T,K) - norm(Deviation(T)) + norm(Kdist(T,K))$$

where $PCL(T,K)$ is the probability of the class label of T w.r.t. the K nearest neighbors (i.e. the frequency of the class label among those K neighbors), $Deviation(T)$ is the sum of distances from all elements from the same class, and $Kdist(T,K)$ is the distance between T and its K nearest neighbors. $norm()$ means normalization. eCODB is now a part of RapidMiner[2]. Following the model from [8], we implemented CB–LOF, a variant of LOF that is capable to manage class labels. We compute CB–LOF (class-based local outlier factor) as an aggregation of two factors, dissimilarity of the case to members of the same class and similarity to members of other classes. As aggregation functions we tested maximum and average. For comparison we use the latter, which performs better in general than maximum.

When compared with RF–OEX, eCODB returned much worse results on the Student solution data. The reason is mainly because of the use of too rough metrics – density and distances – to nearest neighbours and to all members of the class, which does not work well for this 0/1 multidimensional data (number of attributes = 20). Moreover, it is much more difficult to obtain a comprehensive

[2] http://docs.rapidminer.com/studio/operators/data_transformation/data_cleansing/outlier_detection/detect_outlier_cof.html

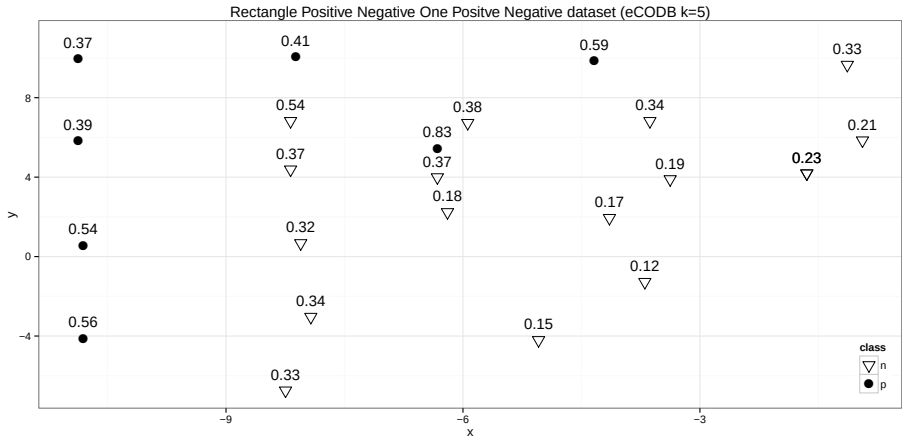

Fig. 3. Results of eCODB (k=5) on an artificial data set; outlier factors are above the datapoints and they are normalized to [0,1] interval, where 1 means the most outlying.

explanation why a particular instance is an outlier. The situation was similar for CB–LOF, although the results were slightly better than with eCODB. For one of the artificial data sets, results of RF–OEX and eCODB can be found in Fig. 2 and 3. More results can be found on www.fi.muni.cz/~popel/685269.

4 Outlier interpretation

As argued elsewhere [1], the outlier factor alone, i.e. the rank of an example, is insufficient for adequate outlier interpretation and explanation. Thus, finding the reason, or reasons, for being an outlier is highly relevant on several application domains. Several methods for constructing an interpretation of outliers have been recently published [1, 5, 14, 15] but only two for class-based outliers.

The first one is CB-ILP [3] briefly described in Section 1. The explanation that CB-ILP offers actually consists of two rule sets - the starting theory (i.e. the rules that have been learned from the full data set) and the ending theory (rules learned after removing an outlier(s)).

The method in [8] analyses frequent patterns that cover an instance/example and takes supports of those patterns for finding the most significant attribute-value couples as an explanation. They define the contradict-ness of an itemset X with respect to a transaction t (in terms of association rule mining) in the following way:

$$Contradictness(X,t) = (card(X) \text{ - } card(t \cap X)) * support(X)$$

The motivation is as follows. The more the itemset X deviates from t, the more contradictory it is. Moreover, the greater the support of X is, the more t deviates from other instances.

We observed that this method gives counter-intuitive results even in very simple situations. The problem is that itemsets containing more items from t can have the same contradict-ness score as itemsets which do not have these items. It is enough if these items occur at least in those transactions in which the other items occur. For example, assume a transaction $t = \{A_1, A_2, ..., A_n\}$, itemsets $X_0 = \{B\}, X_1 = \{B, A_1\}, X_2 = \{B, A_1, A_2\}, ..., X_n = \{B, A_1, A_2, ..., A_n\}$, and $support\{A_1, A_2, ..., A_n\} = 1$. It is clear that support of all these itemsets is the same, i.e. it is equal to $support(X_0)$. Furthermore, it holds for all these itemsets that $card(X)$ - $card(t \cap X) = 1$. Thus, the contradict-ness of all these itemsets is equal to $support(X_0)$. In such a case the contradict-ness score is not appropriate for outlier explanation because items $A_1, A_2, ..., A_n$ do not distinguish t from other instances at all and therefore they are superfluous. Another example, for the ZOO data set, can be found on the supplementary web page. For that drawback we did not use this method.

For class outlier explanation we developed two new methods. Both use already learned random trees and return interpretation of outliers as a set of conjunctions of attributes or attribute-value couples with weights, where a weight is proportional to the expressive power of the conjunction.

4.1 Reduction of random trees

For an outlier, we take all trees that classified this instance into an incorrect class. Actually, we now work with two classes – O as outlier and N as normal – like in the classical outlier detection settings, which allows us to prune the trees, see Fig. 4. Specifically, all subbranches that classify into N can be removed. In the next step, we remove internal nodes in the branch that do not influence classification by checking all values the nodes can have. After this pruning is done, sets of attributes are collected by running outlying instance down each tree. Each of those attribute sets interprets outlierness of the examined point with a weight that is given by the occurrence frequency in the pruned trees. Let us inspect the interpretation of three instances belonging to the most outlying instances in the *iris* data set. The full list of interpretations can be found on the web page.

```
Instance number: 71, Class: Iris-versicolor petalwidth>=1.6, 0.6
Instance number: 84, Class: Iris-versicolor petallength>=4.9, 0.63
Instance number: 37, Class: Iris-setosa sepallength>=5.4 &&
sepalwidth<3.7, 1
```

This method is much more efficient when compared with the ILP approach. However, it prefers short interpretations and sometimes oversees more complex interpretations. The following method is able to find also longer conjunctions.

4.2 Analysis of frequent branches

The second method looks for a frequent combination of attributes, i.e. a combination with support higher than *min_supp*, again on the trees that classify the instance incorrectly. For each frequent combination we express the whole data

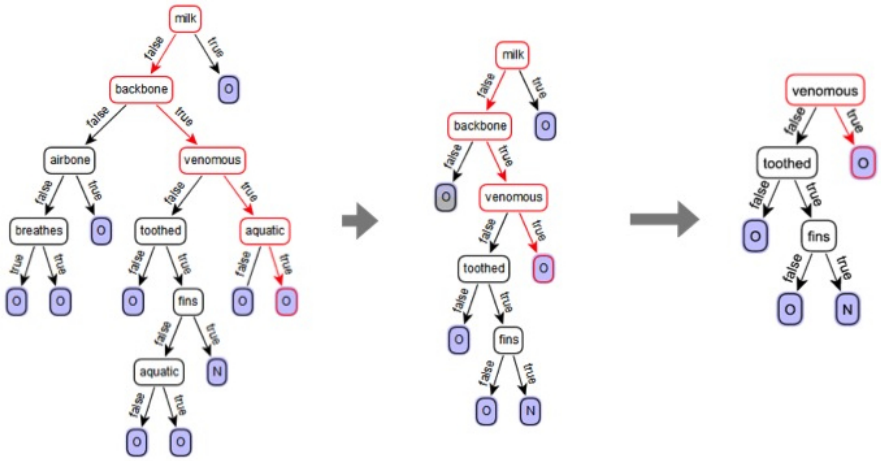

Fig. 4. Tree pruning.

set only by attributes that appeared in that frequent combination and observe how much the outlier factor changed. To compare these two values of the outlier factor, we first have to normalize each one of them. The results are as follows.

```
Instance number: 71, Class: Iris-versicolor petalwidth=1.8, 0.88
```

It means that the outlierness of instance no. 71 is caused from 88% by value 1.8 of attribute *petalwidth*. Now let us have a look at the third most outlying instance number 84:

```
Instance number: 84, Class: Iris-versicolor
petallength=5.1, 0.74
sepallength=6 && petallength=5.1, 0.26
```

Instance outlierness is caused from 74% by the value of *petallength*. There is also a significant increase in outlierness if we combine attribute *petallength* with attribute *sepallength*. This combination participates in outlierness with 26%. Thus, frequent attribute set allows to find more complex interpretation more frequently than the first method. As previously mentioned, supplementary material and results for other data sets can be found on www.fi.muni.cz/~popel/685269.

4.3 Comparison with CB-ILP

For comparison we used Votes data set. The runtime of CB-ILP was 30 minutes, compare to less than a minute of RF–OEX, and it detected 3 negative (republican) and 2 positive (democrat) outliers with gains varying from 0.15 to 0.67. 16 negative examples were labelled as irregularity for being a fact in direct theory as well as 3 positive. There were 7 other positive irregularities, with gain just

Instance	*Class*	*RF-OEX*	*Score*	*eCODB*	*Score*	*ILP*	*Gain*
408	Dem.	1.	36.97	1.	1.31	3. (O.)	1/0.32
7	Dem.	2.	33.21	-	-	2. (O.)	1/0.55
243	Rep.	3.	32.68	-	-	7. (A.)	0.38
268	Rep.	4.	29.32	4.	1.44	1. (O.)	1/0.67
389	Dem.	5.	26.80	-	-	1. (A.)	0.68
394	Rep.	6.	22.69	-	-	-	-
169	Dem.	7.	21.93	-	-	5. (A.)	0.29
108	Rep.	8.	20.50	-	-	-	-

Fig. 5. Comparison-Votes-RF-CODB-ILP.

a little over 0.05. We also detected 6 negative anomalies with gain up to 0.53 and 6 positive anomalies with gain up to 0.68. Detected cases with the highest gain are in Fig. 5 together with results of RF–OEX and eCODB. CB-ILP detected Example 389 (democrat) as anomaly with gain 0.68, because he voted for freezing physician fee, against Synfuels corporation cutback and against duty free exports. Negative example 268 (republican) was identified as outlier with positive gain 1.00

5 Conclusion

In this contribution we argue that outlier detection in labelled data is challenging area both for research and for applications. We brought a review of existing approaches to that problem and introduced a novel method based on Random Forests that is competitive or overcome existing method. Two new methods for class-based outlier description and interpretation were presented and their results were compared with the ILP-based approach.

The open question is evaluation of class-based outlier detection. We performed only a small step in this direction and built several artificial data sets. Building benchmark data for this task more systematically will be the next step. To improve efficiency of RF–OEX, especially its interpretation part, ensemble-based noise elimination and also local models [7] look as good starting points.

Besides the applications mentioned earlier, there are many others that can exploit information about class-based outliers or employ similar techniques, e.g. in the field of subgroup discover [12]. A challenging one is fake text recognition, e.g. an email that pretends to be written by a woman (or a member of a particular group in general), or a similar kind of fake chat contribution.

Now it is up to the reader to answer the question that is in the title of this contribution.

Acknowledgments. We thanks to IDA reviewers for valuable comments and suggestions and to Vaclav Blahut for implementation and experiments with CB-ILP. We would like to thank also to the members of KDLab FI MU for their help. This work has been partially supported by Faculty of Informatics, Masaryk University, Brno.

References

[1] *ODD2 Ws on Outlier Detection & Description under Data Diversity, KDD 2014.*
[2] C. C. Aggarwal. *Outlier Analysis.* Springer, 2013.
[3] F. Angiulli and F. Fassetti. Exploiting domain knowledge to detect outliers. *Data Min. Knowl. Discov.*, 28(2):519–568, 2014.
[4] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.
[5] X. H. Dang, B. Micenková, I. Assent, and R. T. Ng. Local outlier detection with interpretation. In *ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III*, pages 304–320, 2013.
[6] M. Hall et al. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.
[7] B. Frenay and M. Verleysen. Classification in the presence of label noise: A survey. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(5):845–869, May 2014.
[8] Z. He, X. Xu, J. Z. Huang, and S. Deng. Mining class outliers: concepts, algorithms and applications in CRM. *Expert Syst. Appl.*, 27(4):681–697, 2004.
[9] Zengyou He, Shengchun Deng, and Xiaofei Xu. Outlier detection integrating semantic knowledge. In *Advances in Web-Age Information Management*, volume 2419 of *LNCS*, pages 126–131. Springer, 2002.
[10] N. Hewahi and M. Saad. Class outliers mining: Distance-based approach. *International Journal of Intelligent Technology*, 2(1):5568, 2007.
[11] George H. John. Robust decision trees: Removing outliers from databases. In *Knowledge Discovery and Data Mining*, pages 174–179. AAAI Press, 1995.
[12] Rob M. Konijn, Wouter Duivesteijn, Wojtek Kowalczyk, and Arno J. Knobbe. Discovering local subgroups, with an application to fraud detection. In *Proceedings of PAKDD 2013*, pages 1–12, 2013.
[13] Dennis Leman, Ad Feelders, and Arno J. Knobbe. Exceptional model mining. In *ECML/PKDD*, volume 5212 of *LNCS*, pages 1–16. Springer, 2008.
[14] B. Micenková, R. T. Ng, X. H. Dang, and I. Assent. Explaining outliers by subspace separability. In *IEEE ICDM 2013*, pages 518–527, 2013.
[15] E. Müller, F. Keller, S. Blanc, and K. Böhm. Outrules: A framework for outlier descriptions in multiple context spaces. In *ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II*, pages 828–832, 2012.
[16] Spiros Papadimitriou and Christos Faloutsos. Cross-outlier detection. In *Advances in Spatial and Temporal Databases*, volume 2750 of *Lecture Notes in Computer Science*, pages 199–213. Springer Berlin Heidelberg, 2003.
[17] Michael R. Smith and Tony R. Martinez. Improving classification accuracy by identifying and removing instances that should be misclassified. In *IJCNN*, pages 2690–2697. IEEE, 2011.
[18] K. Vaculík, L. Nezvalová, and L. Popelínsky. Educational data mining for analysis of students' solutions. In *AIMSA 2014, Varna, Bulgaria, September 11-13, 2014. Proceedings*, volume 8722 of *Lecture Notes in Computer Science*. Springer.