Outlier and Novelty detection

Clémence Marcaillou, Tudor Oancea, Jean-Baptiste Conan

1 Introduction

Anomaly detection can be defined as the identification of observations, events or items which do not resemble the great majority of the data and which should be treated as *anomalous*.

Needs for anomaly detection arise in many fields of application. A first example could be face emotion recognition software, that for robustness purposes should detect inputs not corresponding to a face in order to avoid spurious predictions. Other examples include new diseases and medical conditions detection, credit card fraud, network intrusion detection, etc.

An important characteristic of anomaly detection is that in practice, the datasets *mainly* contain normal, non anomalous, labeled observations. However, the term 'labeled' is not very appropriate since most anomaly detection algorithms work in an unsupervised fashion. We don't use the labels in the computation, but we still assume the data is *mainly* composed of observation from a single class.

We precised *mainly* because there is an important conceptual distinction to make between *outlier* and *novelty* detection (as outlined in [5]) :

- novelty detection : The dataset is supposed to contain only observations of the normal class and we are trying to build a predictor to detect *future* anomalies.
- outlier detection : The dataset is supposed polluted and we want to find the *pre-existing* anomalies. This can be used for example for cleaning noisy datasets and is usually not characterized as unsupervised learning since there is no actual *learning*.

Hereafter we are going to use *anomaly detection* as a generic term for outlier and novelty detection.

Anomaly detection belongs to the more general subject of *One-Class Classification* (OCC, as extensively described in [3]), that deals with the problem of creating a classification boundary between two classes with data only coming from one of them. The idea is not to separate two classes present in the training data but rather to *surround* the given data with a boundary, i.e. to separate it from 'the rest of the world'.

Another problem that belongs to OCC but should not be confused with anomaly detection is *PU learning* (for Positive-Unlabeled learning). While both can have similar objectives, there are some important practical differences : in PU learning we use labeled data coming from the normal class **and** unlabeled data, not only labeled normal data, and we use the available labels in a *semi-supervised* fashion. Unfortunately we won't be discussing PU learning since it constitutes the subject of another group's project.

In this paper we first present in section 2 some classical algorithms for both outlier and novelty detection and tackle the problem of evaluating their performance. In section 3 we apply these algorithms on concrete data and compare their results.

2 Some classical methods

We chose the following algorithms because they still are popular options even though they all have more mature and optimized variations.

2.1 One class Support Vector Machine (OSVM)

The goal of this method proposed by Schölkopf et. al. in [8] is to find a hyperplane separating the data from the origin and that is as far as possible from the origin, where the anomalous data is supposed to lie. This hyperplane is found with a soft margin formulation similar to classical SVMs. Furthermore, using the kernel trick gives us the possibility to increase the complexity of the constructed frontier and get data points that are always separable from the origin in an appropriate feature space (in the case of RBF kernels).

For n data points $\mathbf{x}_1, \dots \mathbf{x}_n \in \mathbb{R}^p$ this is achieved by solving the following QP :

$$\min_{w \in F, \xi \in \mathbb{R}^n, \rho \in \mathbb{R}} \quad \frac{1}{2}||w||^2 + \frac{1}{\nu n}\sum_i \xi_i - \rho \tag{1}$$

$$\text{s.t.} \quad (w \cdot \phi(\mathbf{x}_i)) \geq \rho - \xi_i, \ \xi_i \geq 0 \tag{2}$$

where $\nu \in (0, 1]$ is a hyperparameter and $\phi : \mathbb{R}^p \to F$ is a feature map to a RKHS F. The final predictor is given by $f(\mathbf{x}) = \text{sign}(w \cdot \phi(\mathbf{x}) - \rho)$.

In practice the QP is solved using the dual problem (whose derivation can be found in the appendix in section 4.1.1) and by taking advantage of its special structure it can be solved more efficiently than a generic QP. There also exists implementations using SGD as the class `SGDOneClassSVM` of `scikit-learn` ([7]).

Interestingly enough, observations in the train set can be found on the wrong side of the constructed hyperplane and the proportion of such outliers is controlled from above by the hyperparameter ν (see 4.1.2 for experimental verification). It controls the 'softness' of the margin and its fine tuning can reduce overfitting in novelty detection procedures, and could even allow us to use this algorithm for outlier detection (in the very restrictive case where we have a good prior idea on the proportion of outliers).

A notable limitation of this algorithm comes from the fact that we arbitrarily assumed that the anomalous class lies around the origin. Other authors proposed to solve this problem by finding a hyperplane that is *attracted* to the center of the data cloud instead of being *repelled* from the origin, which indeed seems more natural. An example of such variation of OSVM can be found in [2].

2.2 Local Outlier Factor (LOF)

This algorithm proposed by Markus M. Breunig et. al. in [1] consists in assigning to each observation an "anomaly score" called the *Local Outlier Factor*. This factor is computed by comparing the local density (called the *Local Reachability Density*) around a given observation x^* and the density around its k-th nearest neighbors using the following procedure:

1. Find the set $N_k(x^*)$ of the k-th nearest neighbors of x^*.
2. Compute the 'distances' of x^* to its neighbors using the quasimetric $\text{rd}_k(x^*, x) := \max\{d_k(x), d(x^*, x)\}$, where $d(x^*, x)$ is the euclidean distance between x and x^* and $d_k(x)$ is the euclidean distance between x and its k-th nearest neighbor.
3. Compute the Local Reachability Density $\text{lrd}_k(x^*)$ (defined in formula 3) as the inverse of the average of these distances.
4. Deduce the Local Outlier Factor $\text{lof}_k(x^*)$ (defined in formula 3)

$$\text{lrd}_k(x^*) := \frac{|N_k(x^*)|}{\sum\limits_{x \in N_k(x^*)} \text{rd}_k(x^*, x)} \qquad \text{lof}_k(x^*) := \frac{\sum\limits_{x \in N_k(x^*)} \frac{\text{lrd}_k(x)}{\text{lrd}_k(x^*)}}{|N_k(x^*)|} \tag{3}$$

In this procedure the set $N_k(x^*)$ can contain more than k elements if there are several observations equidistant to x^*. In the end, the outliers are chosen as the observations with the lowest lof_k (i.e. those that have a density considerably lower than their neighbors). A contamination parameter fixes the number of anomalies that the algorithm will return, meaning that if we set it at 0.01 for example, it will return the 10% observations having the worst LOF. This parameter depends mainly on the context : for a higher precision one can decrease the contamination parameter while one might increase it to improve the recall.

2.3 Isolation Forest (IF)

Isolation Forest first appeared in 2008 and was proposed by Tony Liu in [4] during the *Eighth IEEE International Conference on Data Mining* and is based on random forest. IF has a *training* stage, where it creates a forest of decision trees we call *isolation trees* (or *iTrees*) using random forests algorithms, and an *evaluating* stage, where an anomaly score is computed to identify the anomalies. Intuitively, in the case of outlier detection, the anomalous data (outliers) is geometrically isolated from the majority of the data and needs less steps to be separated from the other data. The outliers would then be the data points with the lowest depth in the iTrees. Conversely, normal data points (inliers) are harder to distinguish from one another and their depth in the iTrees would be bigger.

The anomaly score of an observation x^* is not defined as the average depth $h(x^*)$ of x^* in the iTrees since it depends on the height of each iTree, which can vary a lot. Instead, we want to somehow normalize the depths. To this end, we can see the iTrees as *Binary Search Trees* (BST) and see the path to an external node containing x^* (a leaf) as the path to an unsuccessful search in the BST. We then use the theory of BSTs to compute the anomaly score :

1. We compute the average path length of unsuccessful search in BST : $c(n) = 2H(n-1) - \frac{2(n-1)}{n}$ where $H(k) := \ln(k) + \gamma$ is the approximated k-th harmonic number and n the number of observations.

2. Finally, we define our anomaly score $s(x^*) = 2^{-\frac{\hat{\mathbb{E}}(h(x^*))}{c(n)}}$ with $\hat{\mathbb{E}}(h(x^*))$ the mean of $h(x)$ over the *iTrees* of the forest.

Whenever $s(x^*)$ is close to 1, x^* is more likely to be an anomaly. Conversely, if it is close to 0, it is more likely to be a normal value. For sets without anomalies, all the $s(x^*)$ should be approximately 0.5 with this score definition. Just as in LOF there is a contamination parameter specifying the percentage of observations we want to declare as anomalies.

For novelty detection, the algorithm takes as input the new point to fit and finds the mean path length of this point in the *iTrees* of the isolation forest, to finally compute its score.

2.4 The problem of evaluating anomaly detection algorithms

The evaluation of classifiers is of great importance when comparing the performance of two algorithms on a same dataset or during hyperparameter tuning. Since OCC is close in spirit to traditional binary classification, we are tempted to construct a confusion matrix (by considering the anomalies as the positive class and the normal data as the negative class) and the usual metrics associated (like recall, specificity,...). However, in most cases we do not have all the information we need to evaluate the whole matrix because there are either no examples of anomalies or we don't know which one they are.

Another issue is that most of the metrics used in binary classification are not accurate when dealing with unbalanced classes (cf [3]). Accuracy (defined by $\frac{TP+TN}{P+N}$) for example is not well-suited to anomaly detection since the classifiers may want to perfectly 'learn' all the normal data (and reduce the False Positive rate) and completely ignore the True Positives. The False Negative Rate would be very high, which is undesirable for anomaly detection. There exists however more robust metrics for this kind of unbalanced situations :

- <u>F_1-score</u> : the harmonic mean of the precision (positive predictive value PPV) and the recall (true positive rate TPR) : $F_1 = \frac{2}{1/TPR + 1/PPV} = \frac{2 \times PPV \times TPR}{TPR + PPV}$. Maximizing it corresponds to maximizing both precision and recall.

- <u>AUPRC</u> : the area under the precision-recall curve, it is similar to the area under the ROC curve but AUPRC is preferred for highly unbalanced datasets.

3 Experiments

3.1 USPS dataset

The US Postal Service Dataset is a smaller version of the MNIST dataset of handwritten digits we imported from Kaggle ([6]). It contains 9298 black&white 16x16 images separated in a train set of 7291 images and a test set of 2007 images. There are 256 features corresponding to the intensities of each pixel, and one label giving the digit class (integer between 0 and 9).

We chose this dataset because it is the one originally used by Schölkopf et. al. in their experiments with OSVM in [8], even though today image processing tasks would rather be treated with neural networks. To create novelty detection predictors, we will follow their idea and consider the 0-class as the negative, non anomalous class, train our algorithm on all the 0-class of the train set, and then test it on the whole test set. The 0 digits should be recognized as such and the non-0 should be detected as novelties.

We decided to use the F_1-score in the tuning process because we do not really have a prior on the gravity of anomalies and at what point we have to avoid them.[1] Our tuning process of a generic hyperparameter λ is based on the following variation of the 5-fold cross-validation :

- We split the global train set GT into 5 folds $F_1, \ldots, F_5$ using the class `StratifiedCrossValidation` of `scikit-learn` to ensure that each digit class is evenly spread out between the folds and that each fold has a similar ration of 0s.
- for $i = 1, \ldots, 5$ we isolate the 0-class in F_i to obtain a subset F_{i0} and call F_{-i} the union of the other folds.
- Train the predictor with the current value for λ on F_{i0} and test it on F_{-i} to compute the F_1-score.
- Compute the average F_1-score on all the folds.
- Choose the value for λ that minimizes this average F_1-score.

3.1.1 OSVM

We used the class `OneClassSVM` of `scikit-learn` to create a novelty predictor. We avoided tuning the kernel bandwidth γ by using its option `'scale'` and focused on the tuning of the parameter ν described in 2.1 and 4.1.2.

Using the procedure described above in 3.1 we obtained an optimal average CV F_1-score of 0.985 attained for $\nu = 0.1$. The whole results can be found in table 5. After retraining on the whole GT and testing on the test set, we obtained a slightly lower F_1-score of 0.978. The full results can be seen in Figure 1 and Table 1. We notice in particular that, as we would desire, there are very few False Negatives, i.e. non detected novelties.

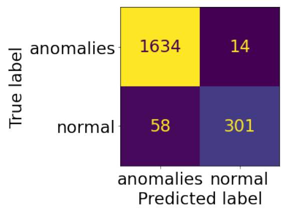

Figure 1: Confusion matrix for tuned OSVM

F_1-score	0.978
precision score	0.966
recall score	0.992
accuracy score	0.964

Table 1: Values for different metrics of our tuned OSVM predictor

[1]The F_1-score does not prioritize recall over precision or vice-versa so optimizing it tries to find a trade off between TPs and TNs.

3.1.2 IF

For IF we used the class `IsolationForest` of `scikit-learn.ensemble`. We avoided tuning the different parameters by using its option `'auto'` and focused on the tuning of the parameter `n_estimators`.

Using the same procedure described in 3.1 we obtained an optimal average CV F_1-score of 0.977 attained for `n_estimators`=192. The whole results can be found in the graph 5. After retraining on the whole *GT* and testing on the test set, we obtained a slightly lower F_1-score of 0.965. We notice in particular that, as we would desire, there are very few False Negatives, i.e. non detected novelties.

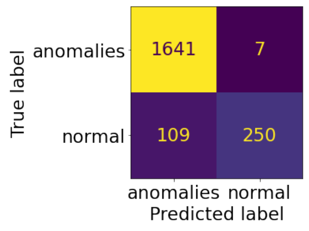

Figure 2: Confusion matrix for tuned IF

F_1-score	0.965
precision score	0.942
recall score	0.996
accuracy score	0.937

Table 2: Values for different metrics of our tuned IF predictor

3.2 Credit Card dataset

This dataset [9] consists of 200 000 transactions made by credit card owners in September 2013. The features consists in 28 variables resulting from a PCA transformation, an *Amount* variable and a *Time* variable, but no more background information about the data is provided for confidentiality purposes. Lastly, a *Class* feature takes value 1 in case of fraud and 0 otherwise. It is highly unbalanced as the frauds accounts for only 0.172% of all transactions.

Given the class imbalance ratio, we decided to use the AUPRC (area under the precision-recall curve) as the preferred metric for hyperparameter tuning. In both methods we first extract a 'training set' consisting of 100 000 observations from the dataset. Neither LOF nor IF really need a 'training' set since they both fit directly on the 'test set' but it is useful to first try the algorithm on a training set using the labels to set the hyperparameters. We then extract a 'test set' of 25 000 observations from the remaining observations in the data set in order to test each model and measure its efficiency.

3.2.1 LOF

We used the `LocalOutlierFactor` from `scikit-learn` and decided to initially fix the contamination parameter at 0.004 which is above the real contamination in the training set (0.0019) because in our case false negatives should definitely be avoided. To choose the number of neighbors, we split the training test into 4 parts and for each part, we compared the AUPRC score for multiple values of k. For each set, we kept the best number of neighbors (respectively 75, 85, 90 and 100) and computed the mean which is equal to 87. We observed that the best k values are not too scattered so we expect that the result will generalize well. Note that it is important that the evaluation of the number of neighbors is done on sets that have the same size as the test set size because this parameter is not independent of the size of the dataset. Then we compared the results for different contamination values (see figure 6 for details) and decided that the best value was 0.004. The final results on the test set are given in figure 3 and table 3.

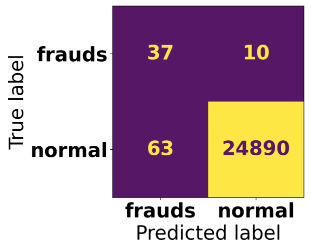

Figure 3: Confusion matrix for tuned LOF

F_1-score	0.503
precision score	0.37
recall score	0.787
accuracy score	0.997
AUPRC	0.557

Table 3: Values or different metrics of our tuned LOF predictor

3.2.2 IF

As for the USPS, we used the `scikit-learn` library. When we tried changing the different parameters, we observed that the only one that had a significant impact on the AUPRC was the `n_estimators` parameter. For `n_estimators`, best AUPRC occurred around 100, so we decided to fix this parameter at this value.

For the contamination parameter, values above 0.02 does not increase the number of detected fraud anymore (see 7), so we shall stay under this range. Moreover, for small contamination such as 0.004, we achieve great precision but many frauds are missed. We must therefore have a trade off, and as we choose to ensure most of the frauds are detected, 0.015 value for the contamination parameter seems to be a good candidate.

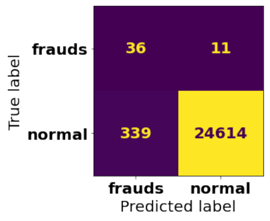

Figure 4: Confusion matrix for tuned IF

F_1-score	0.170
precision score	0.096
recall score	0.766
accuracy score	0.986
AUPRC	0.25

Table 4: Values or different metrics of our tuned IF predictor

3.3 Discussion of the results

For the USPS dataset, OSVM and IF algorithms yield comparable results. The advantage of IF is its linear time complexity compared to the quadratic complexity of OSVM. Besides, we observe that for the credit card dataset, the different metrics tend to stipulate that LOF performs better than IF in finding outliers. A drawback of LOF compared to IF is its quadratic complexity. Both algorithm achieve a good recall, which means that most outliers are detected, which was the desired behavior. We also notice that the accuracy is always quite high, although the recall results on outlier detection are not that good.

Conclusion

To sum up, anomaly detection is a complicated field of study that is still the object of active research. With the years more and more mature algorithms are published and methods for tackling the issue of evaluation of classifiers are discovered.

Unfortunately we only we briefly discussed the problem of model tuning. We want to emphasize that this process varies a lot with depending on the objectives and the importance one attaches to anomalies. In the context of novelty detection, is it more important to detect all novelties at the expense of many false alerts, or to find as many as possible with a minimum of false alerts ? In the context of outlier detection, is the algorithm intended to be used alone or in combination with others? Knowing the cost of missing an anomaly versus the cost of misclassifying an observation as an anomaly can help tackle this issue. Usually these costs are considered equal but this is an oversimplification and that's why there is a need to develop cost-sensitive methods that are not dependent on the training data.

4 Appendix

References

[1] Markus M. Breunig et al. *LOF: Identifying Density-Based Local Outliers.* URL: `https://www.dbs.ifi.lmu.de/Publikationen/Papers/LOF.pdf`.

[2] Colin Campbell and Kristin Bennett. "A Linear Programming Approach to Novelty Detection". In: *Advances in Neural Information Processing Systems.* Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press, 2001. URL: `https://proceedings.neurips.cc/paper/2000/file/0e087ec55dcbe7b2d7992d6b69b519fb-Paper.pdf`.

[3] Shehroz S. Khan and Michael G. Madden. "One-Class Classification: Taxonomy of Study and Review of Techniques". In: *CoRR* abs/1312.0049 (2013). arXiv: `1312.0049`. URL: `http://arxiv.org/abs/1312.0049`.

[4] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation Forest". In: *2008 Eighth IEEE International Conference on Data Mining.* 2008, pp. 413–422. DOI: `10.1109/ICDM.2008.17`. URL: `https://ieeexplore.ieee.org/abstract/document/4781136`.

[5] *Novelty and Outlier detection.* URL: `https://scikit-learn.org/stable/modules/outlier_detection.html`.

[6] IEEE Transactions on Pattern Analysis and Machine Intelligence. "USPS Dataset". In: (2018). URL: `https://www.kaggle.com/bistaumanga/usps-dataset`.

[7] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[8] Bernhard Schölkopf et al. "Estimating the Support of a High-Dimensional Distribution". In: *Neural Comput.* 13.7 (July 2001), pp. 1443–1471. ISSN: 0899-7667. DOI: `10.1162/089976601750264965`. URL: `https://doi.org/10.1162/089976601750264965`.

[9] Machine Learning Group - ULB. *Credit Card Fraud Detection dataset.* Anonymized credit card transactions labeled as fraudulent or genuine. 2018. URL: `https://www.kaggle.com/mlg-ulb/creditcardfraud`.

4.1 More on OSVM

4.1.1 Dual problem

The QP 1 is hardly solvable because the parameters belong to an infinite dimensional Hilbert space. By using Lagrange multipliers α_i, β_i we can form the Lagrangian

$$\mathcal{L}(w, \xi, \rho, \alpha, \beta) = \frac{1}{2}||w||^2 + \frac{1}{\nu l}\sum_i \xi_i - \rho - \sum_i \alpha_i((w \cdot \phi(\mathbf{x}_i)) - \rho + \xi_i) - \sum_i \beta_i \xi_i$$

We can set its partial derivatives with respect to the primal variables $w, \xi\rho$ to 0 to get

$$\begin{cases} w = \sum_i \alpha_i \phi(\mathbf{x}_i) \\ \alpha_i = \frac{1}{\nu l} - \beta_i \leq \frac{1}{\nu l} \\ \sum_i \alpha_i = 1 \end{cases}$$

And under this constraints the Lagrangian becomes

$$\begin{aligned}\mathcal{L}(w,\xi,\rho,\alpha,\beta) &= \frac{1}{2}\Big\langle \sum_i \alpha_i \phi(\mathbf{x}_i), \sum_j \alpha_j \phi(\mathbf{x}_j)\Big\rangle + \frac{1}{\nu l}\sum_i \xi_i - \rho - \sum_i \alpha_i \Big\langle \sum_j \alpha_j \phi(\mathbf{x}_j), \phi(\mathbf{x}_i)\Big\rangle \\ &+ \rho \sum_i \alpha_i - \sum_i \alpha_i \xi_i + \sum_i (\alpha_i - \frac{1}{\nu l})\xi_i \\ &= -\frac{1}{2}\sum_{i,j} \alpha_i \alpha_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle = -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j)\end{aligned}$$

where k is the kernel function associated with the feature map. Then the dual QP problem reads

$$\begin{aligned}\min_{\alpha\in\mathbb{R}^n} \quad & \frac{1}{2}\sum_{ij}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j) \\ \text{s.t.} \quad & 0 \le \alpha_i \le \frac{1}{\nu l},\ \sum_i \alpha_i = 1\end{aligned}$$

We notice that this problem is in particular finite-dimensional and can thus be easily solved with any QP solver.

4.1.2 Further experimental results on OSVM

As proved in [8], the hyperparameter ν verifies

$$\text{fraction of outliers} \le \nu \le \text{fraction of support vectors (SVs)} \tag{4}$$

In table 5 you can find empirical values for all those quantities we determined by fitting OSVM on the train set *GT* of the USPS dataset and predicted the labels of the same data. We can see that the inequalities 4 are not always verified. This can come from the fact that the solution found for the optimisation problem is not necessarily an exact solution.

ν	average F_1-score	fraction of outliers	fraction of SVs
0.01	0.980327	0.025963	0.055276
0.02	0.979916	0.030988	0.056951
0.03	0.980089	0.037688	0.059464
0.04	0.980861	0.044389	0.066164
0.05	0.981379	0.051089	0.070352
0.06	0.982412	0.060302	0.077052
0.07	0.983052	0.072027	0.089615
0.08	0.982926	0.081240	0.096315
0.09	0.983862	0.089615	0.106365
0.10	0.984870	0.100503	0.113903
0.20	0.979280	0.197655	0.210218
0.30	0.970784	0.301508	0.309883
0.40	0.961379	0.399497	0.407873
0.50	0.952587	0.500838	0.507538
0.60	0.943969	0.599665	0.603015
0.70	0.935281	0.700168	0.701843
0.80	0.927455	0.799832	0.801508
0.90	0.919339	0.898660	0.900335

Table 5: Experimental values obtained during the cross-validation of OSVM on the **USPS dataset** for 18 values of ν

4.2 Further results of IF on the USPS dataset

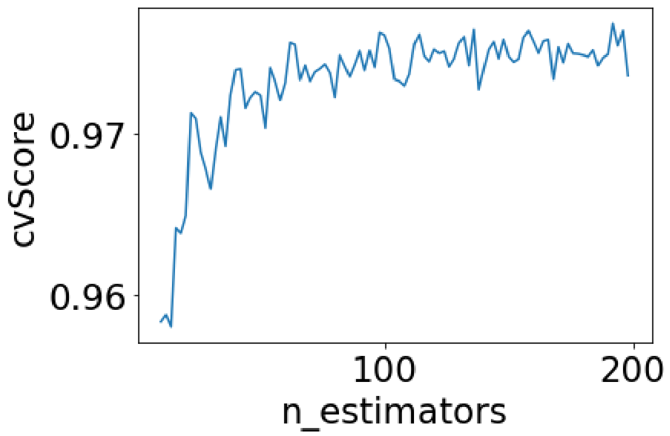

Figure 5: Average F_1-score obtained during the cross-validation of IF on the **USPS dataset** for 100 values of n_estimators

4.3 Comparison of the metrics for different contamination values for the credit card dataset

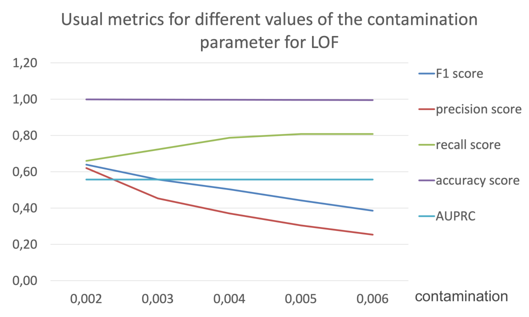

Figure 6: A comparison of the LOF algorithm's performance on the **credit card dataset** when we vary the contamination parameter

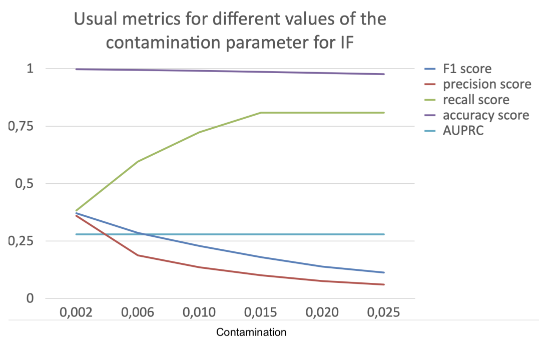

Figure 7: A comparison of the IF algorithm's performance on the **credit card dataset** when we vary the contamination parameter