INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING, *SERIES B*
Volume 5, Number 3, Pages 238–254

USING SELF-ORGANIZING MAPS FOR BINARY CLASSIFICATION WITH HIGHLY IMBALANCED DATASETS

VINICIUS ALMENDRA AND DENIS ENĂCHESCU

Abstract. Highly imbalanced datasets occur in domains like fraud detection, fraud prediction, and clinical diagnosis of rare diseases, among others. These datasets are characterized by the existence of a prevalent class (e.g. legitimate sellers) while the other is relatively rare (e.g. fraudsters). Although small in proportion, the observations belonging to the minority class can be of a crucial importance. In this work we extend an unsupervised learning technique – Self-Organizing Maps – to use labeled data for binary classification under a constraint on the proportion of false positives. The resulting technique was applied to two highly imbalanced real datasets, achieving good results while being easier to interpret.

Key words. unsupervised learning, self-organizing maps, imbalanced datasets, supervised learning

1. Introduction

Highly imbalanced datasets arise in several real-world machine learning problems. One example is fraud detection and prediction at online auction sites. In order to keep their business growing, online auction sites like eBay need to protect buyers from unscrupulous sellers. Among the several types of fraudulent behavior that take place in online auction sites, the most frequent one is non-delivery fraud [1, 2]. The challenge faced by site operators is to identify fraudsters *before* they strike, in order to avoid losses due to unpaid taxes, insurance, badmouthing etc. In other words, for a given product listing they need to *predict* whether or not it will end up being a fraud case, in order to prevent damage. Another example is the clinical diagnosis of relatively rare but serious diseases. In this situation a false negative (someone who has the disease diagnosed as sane) is more damaging that a false positive. An important challenge when tackling these problems is the difficult interpretation of many supervised learning models. The importance of interpretable models lies in the fact that prediction in the above-stated problems is a delicate issue: usually one cannot take harsh measures against a seller just because a model predicts his listing is fraudulent, unless there is a high degree of confidence. The same holds in other domains, where false positives imply additional costs and risks. The decision to take some measure is much easier when stakeholders understand why an observation was labeled as positive.

In this paper we present an algorithm that combines an unsupervised clustering technique – the Self-Organizing Map – with the supervised learning paradigm through the use of labeled data. The proposed algorithm tackles the problem of binary classification for highly imbalanced data under a constraint on the number of false positives. Using labeled data to automatically identify clusters of observations with high probability of being positive makes the Self-Organizing Map an useful tool for exploratory data analysis, which helps understanding the data. Although we developed the proposed method focusing on fraud prediction at online auction

Received by the editors February 17, 2014.
2000 *Mathematics Subject Classification.* 62H30.

sites, it can also be used in other domains. This paper is an extended version of a previously published work [3].

In Section 2 we will present the context for our research; in Section 3 we succinctly describe Self-Organizing Maps; in Section 4 we will explain our proposed algorithm for binary classification for highly imbalanced datasets; in Section 5 we will present the experimental results, and in Section 6 we will discuss them.

2. Related Work

Two problems were highly imbalanced datasets arise are *fraud detection and prediction*. Bolton and Hand [4] did a comprehensive review regarding statistical fraud detection in several domains: credit card fraud, money laundering, telecommunications fraud, computer intrusion, and scientific fraud. Although they did not mention fraud at online auction sites, these challenges also apply. There are recently published papers specifically focused on fraud at online auction sites, some from a descriptive perspective [5–7], and others aiming fraud prediction [8–15]. Regarding the methods employed, the majority of existing works were based on supervised learning techniques: decision trees [9,10], Markov random fields [12], instance-based learners [8], logistic regression [13], online probit models [14], Adaptive Neuro-Fuzzy Inference System [16], Boosted trees [15].

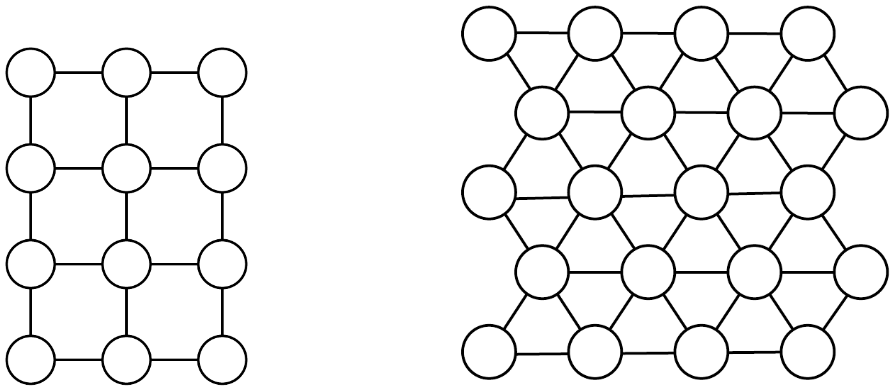
Unsupervised learning methods are also used for fraud detection [4, 17]. Zaslavsky et al. [18] used Self-Organizing Maps for credit card fraud detection, creating maps to capture the typical transaction patterns and then using these maps to classify new transactions. A given transaction is classified as suspicious if the distance to its cell's weight vector is beyond some threshold. Quah et al. [19] proposed a similar system. In essence their approach is one of fraud detection through outlier identification, which does not translate directly to fraud prediction, since they are dealing with fraudulent behavior *that already happened.* The fraud prediction problem deals with identifying who might commit fraud, which is a much harder problem, since fraudsters actively try to disguise themselves as legitimate users. This is crucial in the domain of online auction sites, since fraudsters need to convince buyers (and the online auction site) of their honesty. Therefore, we do not expect a clear outlier pattern in this case.

In a previous work we adapted a technique for exploratory data analysis – Andrews curves – in order to do supervised learning [20]. In the present work we follow a similar strategy: we combine the power of Self-Organizing Maps – an unsupervised learning technique with good visualizations and thus ensuring the understandability of the results – with labeled data in order to classify new examples.

Regarding class imbalance, some common approaches to solve this problem are undersampling of the majority class, oversampling of the minority class, and SMOTE (Synthetic Minority Over-sampling Technique) [21]. Some of the above-mentioned works used undersampling [8, 9], one uses an unsupervised model [12], others did not state the approach adopted [11, 13, 14].

3. Self-organizing Maps

The Self-Organizing Map [22] is a feed-forward neural network whose units are linear and topologically ordered in a bidimensional lattice of a given size. It is a model inspired in the several types of "maps" that exist in the brain of higher animals, linking for example the skin sensations of the different body portions to specific areas in the cortex [22]. Figure 3.1 depicts two examples of map topology: one with a grid topology and another with an hexagonal one.



(A) Grid topology (B) Hexagonal topology

FIGURE 3.1. Examples of Self-Organizing maps' topologies

Each unit i has an weight w_i of the same dimension of input data. All weights are initialized with random values. The network is continuously updated taking one input example at a time (chosen randomly) – x_k – and applying the following algorithm:

- The winning unit i^* is selected as the one that minimizes the euclidean distance $\|x_k - w_i\|$ – a competitive learning approach;
- The weights of units are updated according to the following rule:

$$w_i = w_i + \rho \cdot \Phi(i, i^*) \cdot (x_k - w_i) \tag{3.1}$$

where ρ is the learning rate and $\Phi()$ is the neighboring function, which is a monotonically decreasing function of the distance between units i and i^* in the given network topology. Informally speaking, each new input "attracts" the weight of the winner unit and also of its neighbors, and the degree of attraction grows with the closeness. This closeness can be given e.g. by the link distance: the number of connections in the shortest path between the two given units. The smaller the link distance, the closer are the units. Both the learning rate and the neighboring function decrease over time, in order to guarantee convergence. With this algorithm similar inputs will tend to be assigned to units that are topologically close in the map and the distance between the weight vectors of these units will reflect the distance between the examples assigned to them. This leads to some important properties of Self-Organizing Maps: they capture both density and topology of input examples and map them in a bidimensional representation. The trained network can be visualized in many different ways: number of observations assigned to each unit, distance between units, weights planes, weights positions etc. Once trained, the network can be used to cluster new data: a new observation is assigned to the cell whose weight vector is closest to it.

4. Binary Classification Using Self-Organizing Maps

In this Section we propose a two-phase algorithm for binary classification of high imbalanced datasets. This algorithm combines clustering using Self-Organizing Maps with additional steps to label observations. For sake of brevity, from now on we will write Self-Organizing Map as SOM. We will refer to the observations of the minority class as the *positive* observations, while the others will be the *negative* observations. In the fraud prediction problem, the positive observations are the fraudulent listings and the negative observations are the legitimate listings.

4.1. Using Clustering for Binary Classification. SOM itself is an unsupervised learning technique: it simply clusters observations using the algorithm described in Section 3. To use it for supervised learning, we followed this general procedure:

(1) use training data to identify cluster centers (weights). This is the *training* of the SOM map. From now on we will use the term *cluster* to refer to a SOM's unit;
(2) cluster all training data using the calculated clusters' centers. This means finding for each training observation which is the closest cluster center;
(3) use training labels to *label the clusters* based on the distribution of training data in the clusters;
(4) cluster new data using calculated clusters' centers;
(5) label each new observation *according to the label of its cluster.*

Our two-phase classification algorithm for high imbalanced datasets consists of the sequential application of two distinct classifiers; both use the above-described procedure, but in different ways. The first classifier labels observations as either negative or positive, but observations labeled as positive are sent to the second classifier, which labels them again as negative or positive, giving the final classification.

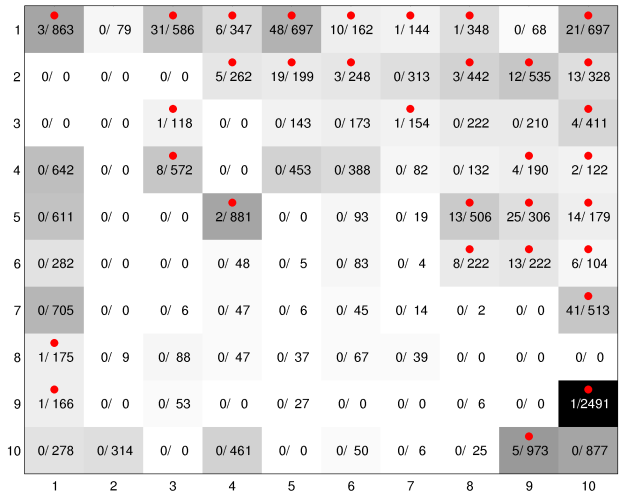
4.2. Phase 1: Filtering Out Negative Observations. As we have already done in a previous work [23], we propose the use of a *filtering phase*, which consists in a classifier with very high true positives rate (recall). Our objective is to generate a labeling with the following characteristics: (i) almost all positive observations are (correctly) classified as such, and (ii) a substantial part of the negative observations is (correctly) classified as negative. The observations classified as positive need further processing with another classifier, which will have the advantage that this new set will be less imbalanced.

This phase follows the procedure described in the previous section with the changes below:

- In step 1: SOM clusters' centers are calculated using only negative observations, since we want to set apart positive observations from the negative ones. The rationale is to reduce the need of retraining, since the distribution of negative observations in the feature space varies much slower than the distribution of positive observations, at least in a fraud prediction scenario, and positive observations bring a very little contribution due to imbalance. Some early experiments showed a slight advantage when using this solution.
- In step 3: a cluster is labeled as *positive* if in the training data assigned to it at least one observation belongs to the positive class.

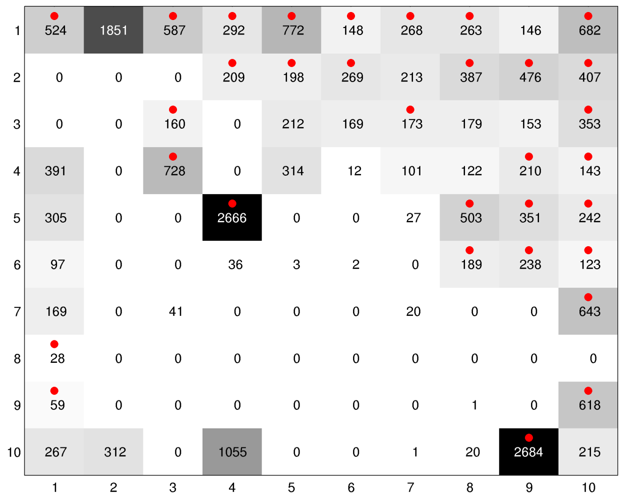
In Figure 4.1 we illustrate this process: Figure 4.1a shows a trained map with the training observations assigned to its clusters and the cluster labeling resulting from step 3 of procedure. Figure 4.1b shows the same map but this time applied to unlabeled data – result of the step 5. In this concrete example, 33% of the observations were labeled as negative, sending the remaining 67% to the next phase.

4.3. Phase 2: Classification with Constraint on False Positives. The second phase gives the final label to the observations coming from the filtering phase.



	1	2	3	4	5	6	7	8	9	10
1	● 3/ 863	0/ 79	● 31/ 586	● 6/ 347	● 48/ 697	● 10/ 162	● 1/ 144	● 1/ 348	0/ 68	● 21/ 697
2	0/ 0	0/ 0	0/ 0	● 5/ 262	● 19/ 199	● 3/ 248	0/ 313	● 3/ 442	● 12/ 535	● 13/ 328
3	0/ 0	0/ 0	● 1/ 118	0/ 0	0/ 143	0/ 173	● 1/ 154	0/ 222	0/ 210	● 4/ 411
4	0/ 642	0/ 0	● 8/ 572	0/ 0	0/ 453	0/ 388	0/ 82	0/ 132	● 4/ 190	● 2/ 122
5	0/ 611	0/ 0	0/ 0	● 2/ 881	0/ 0	0/ 93	0/ 19	● 13/ 506	● 25/ 306	● 14/ 179
6	0/ 282	0/ 0	0/ 0	0/ 48	0/ 5	0/ 83	0/ 4	● 8/ 222	● 13/ 222	● 6/ 104
7	0/ 705	0/ 0	0/ 6	0/ 47	0/ 6	0/ 45	0/ 14	0/ 2	0/ 0	● 41/ 513
8	● 1/ 175	0/ 9	0/ 88	0/ 47	0/ 37	0/ 67	0/ 39	0/ 0	0/ 0	0/ 0
9	● 1/ 166	0/ 0	0/ 53	0/ 0	0/ 27	0/ 0	0/ 0	0/ 6	0/ 0	● 1/2491
10	0/ 278	0/ 314	0/ 0	0/ 461	0/ 0	0/ 50	0/ 6	0/ 25	● 5/ 973	0/ 877

(A) Trained SOM for the filtering phase. Clusters with at least one positive observation were labeled as positive.



	1	2	3	4	5	6	7	8	9	10
1	● 524	1851	● 587	● 292	● 772	● 148	● 268	● 263	146	● 682
2	0	0	0	● 209	● 198	● 269	213	● 387	● 476	● 407
3	0	0	● 160	0	212	169	● 173	179	153	● 353
4	391	0	● 728	0	314	12	101	122	● 210	● 143
5	305	0	0	● 2666	0	0	27	● 503	● 351	● 242
6	97	0	0	36	3	2	0	● 189	● 238	● 123
7	169	0	41	0	0	0	20	0	0	● 643
8	● 28	0	0	0	0	0	0	0	0	0
9	● 59	0	0	0	0	0	0	1	0	● 618
10	267	312	0	1055	0	0	1	20	● 2684	215

(B) Trained SOM for the filtering phase applied to the new data. All observations outside positive clusters are labeled as negative.

FIGURE 4.1. 10×10 SOM for filtering phase. Each cluster contains the number of positive/negative observations assigned to it. Clusters with a spot are the ones labeled as positive. Cell background indicates the proportion of observations assigned to it.

As such it can be seen as an ordinary classification task, but with one added aspect: the enforcement of the *false positives rate*. Fraud prediction and detection usually have a trade-off regarding true versus false positives [4]. Following the same approach of a previous work [15], we assume the existence of a constraint in the proportion of false positives – FP_{max}. In the context of fraud prediction, an online auction site might tolerate $FP_{max} = 15\%$, while other one might tolerate $FP_{max} = 25\%$.

To enforce this restriction, the general procedure presented in Section 4.1 is adapted in the following way:

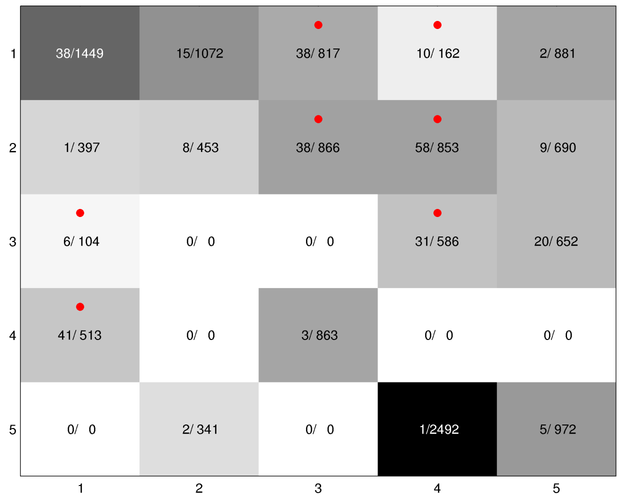
- In step 1: SOM clusters' centers are calculated using only negative observations, similarly to phase 1;
- In step 3: a cluster is labeled as *negative* if in the training data assigned to it all observations belong to the negative class; otherwise, it is temporally labeled as *undecided* and is further processed using the algorithm described below, in order to give the final label.

The undecided clusters have mixed observations, so labeling one of them as positive necessarily increases both false positives and true positives (and vice versa), although these changes are usually different, since some clusters have proportionally more positive observations then others. Given that we have a discrete set of clusters to choose and each one has a discrete set of observations, we need to solve an optimization problem: finding the set of clusters that, if labeled as positive, *maximizes the number of true positives for the given maximum acceptable number of false positives.* Posed this way, our problem can be reduced to the classical 0-1 knapsack problem: given a set of items, each one with a weight and a value, find the best subset of items in terms of total value for a given maximum weight, with the restriction that each item can appear at most once. In our case:

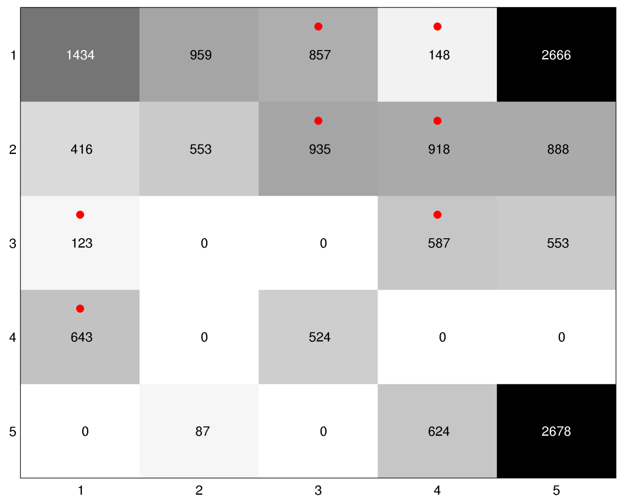
- The individual items are the clusters of the SOM;
- The weight of each cluster is the number of negative observations assigned to it;
- The value of each cluster is the number of positive observations assigned to it;
- The maximum weight is the maximum acceptable number of false positives, which is the number of negative observations in the training set times FP_{max}.

The 0-1 knapsack problem is NP-hard but for the instances that we are dealing this does not matter. We opted to use the classical algorithm based on dynamic programming due to its simplicity. This algorithm is $O(nW)$, where n is the number of items and W is the length in bits of the maximum weight. The maximum number items (clusters in our case) is bounded by the number of positive observations in the training set, which is by definition proportionally small. The maximum weight depends on the number of negative observations used for training, which does not need to grow unbounded. So the running time will end up being dominated by SOM's training time. Solving the 0-1 knapsack problem yields the optimal labeling of clusters for the given FP_{max}, which is used in the steps 4 and 5 of the general algorithm to label the remaining observations.

In Figure 4.2 we illustrate this process, similarly to what was done in Figure 4.1. In Figure 4.2a one can see that some clusters with positive observations were not labeled as positive, so as to not violate the maximum false positives limits.



(A) Trained SOM for classification phase after step 3, with clusters labeled based on the constraint $FP_{max} \leq 20\%$.



(B) Trained SOM for classification phase applied to the new data (step 4 and 5). All observations inside the positive clusters are labeled as positive, while the others are labeled as negative.

FIGURE 4.2. 5×5 SOM for phase 2. Each cluster displays the number of positive/negative assigned to it. Clusters with a spot are the ones labeled as positive (i.e. chosen to maximize true positives under the constraint FP_{max}). Cluster background indicates the proportion of observations assigned to it.

4.4. Final Algorithm. Although conceptually the two phases are run sequentially, in practice there is some interleaving, since it is not necessary to retrain the SOMs every time. The real sequence is the following:

(1) Training (done once for one training set): *steps 1–3 of filtering phase* $\rightarrow$ *steps 1–3 of classification phase*
(2) Applying: *steps 4–5 of filtering phase* $\rightarrow$ *steps 4–5 of classification phase*

Algorithm 1 shows the pseudocode for steps 1–3 of filtering phase, and Algorithm 2, the same for classification phase. Algorithm 3 contains the steps 4 and 5 of both phases. Figure 4.3 portraits the graphically the different algorithm phases and their connections.

The final algorithm has three parameters: besides FP_{max}, it also needs the sizes of the two SOMs. These two parameters can be selected through cross-validation with the training set, in order to find the pair that maximizes the true positives rate.

```
Input: A $n \times p$ matrix $\mathbf{X}$ with training data
Input: A vector $y$ of length $n$ with labels (0 or 1)
Input: Map dimension $d_1$
Input: $FP_{max}$
Output: trained SOM map and list of positive clusters
// Step 1: calculate cluster centers
$\mathbf{X}_{neg} \leftarrow \{\mathbf{X}[i,:] | y[i] = 0\}$;
$som1 \leftarrow$ train_som($\mathbf{X}_{neg}, d_1$);
// Step 2: assign data to clusters
$som1\_out \leftarrow$ apply_som($som1, \mathbf{X}$);
// Step 3: labels as positive clusters with positive
   observations
$p\_clusters1 \leftarrow \{\}$;
for $i \leftarrow 1$ to num_clusters($som1\_out$) do
    $obs \leftarrow som1\_out[i].observations$;
    $num\_positives \leftarrow \sum_{j \in obs} y[j]$;
    $num\_negatives \leftarrow$ length($obs$) $- num\_positives$;
    if $num\_positives > 0$ then
        $p\_clusters1 \leftarrow p\_clusters1 \cup i$
    end
end
// Separate positive observations for phase 2
$\mathbf{X_2} \leftarrow \mathbf{X}[som1\_out[p\_clusters1].observations, :]$;
$y_2 \leftarrow y[som1\_out[p\_clusters1].observations]$;
```

Algorithm 1: Steps 1–3 of phase 1 (filtering out negative observations)

5. Experimental Results

5.1. Dataset Description.

5.1.1. ML dataset. The dataset used consists of 43,775 product listings extracted from MercadoLivre (www.mercadolivre.com.br), the biggest Brazilian online auction site. For each listing we collected 11 features about the product itself

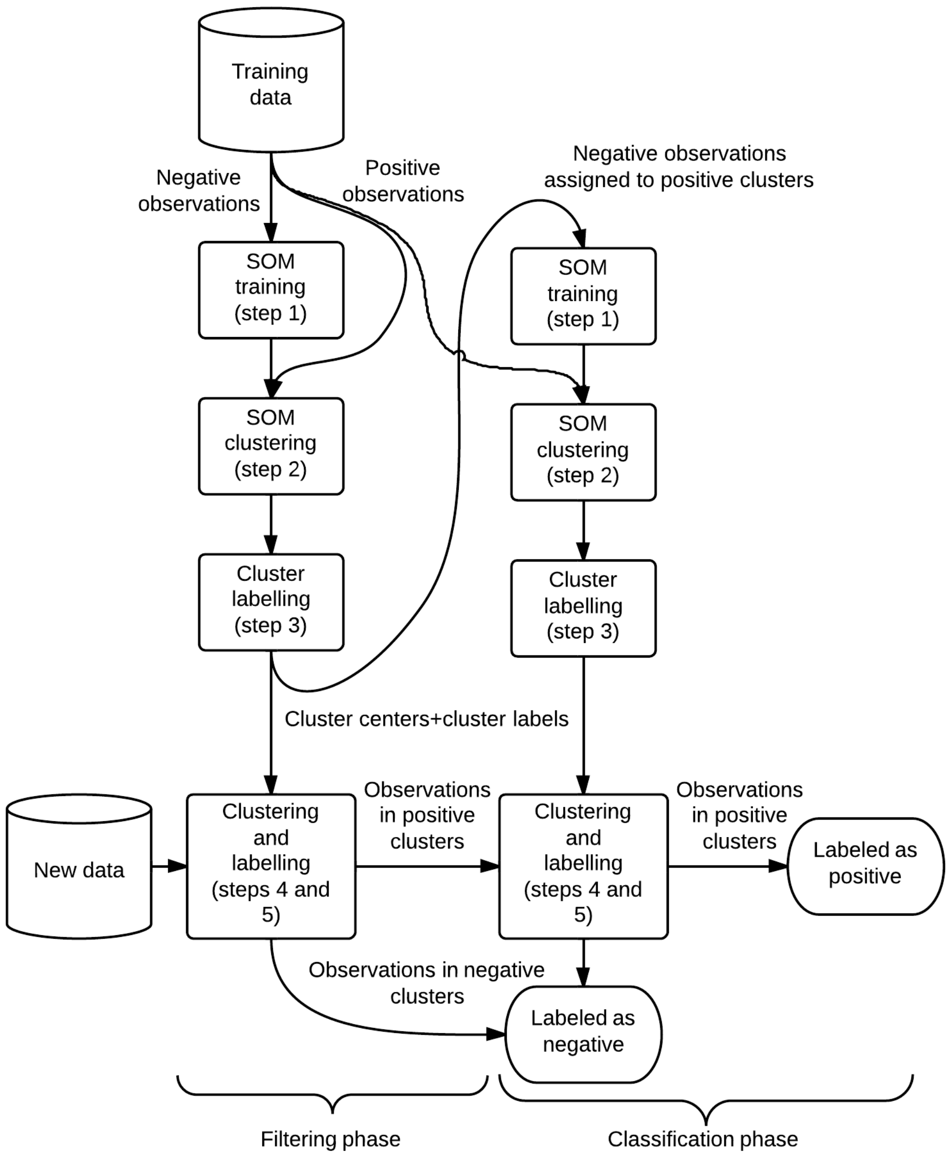

Training data
Negative observations
Positive observations
Negative observations assigned to positive clusters
SOM training (step 1)
SOM clustering (step 2)
Cluster labelling (step 3)
SOM training (step 1)
SOM clustering (step 2)
Cluster labelling (step 3)
Cluster centers+cluster labels
New data
Clustering and labelling (steps 4 and 5)
Observations in positive clusters
Clustering and labelling (steps 4 and 5)
Observations in positive clusters
Labeled as positive
Observations in negative clusters
Labeled as negative
Filtering phase
Classification phase

FIGURE 4.3. Schematic view of classification algorithm

(price, relative price difference), about its seller (reputation, account age, number of recent transactions) and about the product's category (number of listings, number of sellers, average listing price, number of listings at level 3, number of sellers at level 3, Category fraud rate at level 2). We collected information about listings very close to the moment they were published, therefore before any sign of fraudulent behavior had appeared. This is important since we wanted to evaluate fraud *prediction* algorithms. One percent of the listings (439) evolved over time to highly suspicious situations (several negative feedback mentioning non-delivery fraud and account suspension), so we labeled these as fraudulent listings, while

```
Input: A $n \times p$ matrix $\mathbf{X_2}$ with training data coming from phase 1
Input: A vector $y_2$ of length $n$ with labels (0 or 1)
Input: Map dimension $d_2$
Input: $FP_{max}$
Output: trained SOM map, list of positive clusters
// Step 1:  calculate cluster centers
$\mathbf{X_{neg_2}} \leftarrow \{\mathbf{X_2}[i,:] | y_2[i] = 0\}$;
$som2 \leftarrow$ train_som($\mathbf{X_{neg_2}}, d_2$);
// Step 2:  assign training data to clusters
$som2\_out \leftarrow$ apply_som($som2, \mathbf{X_2}$);
// Step 3:  choose positive clusters using knapsack
$clusters, weights, values \leftarrow []$;
$k \leftarrow 0$;
for $i \leftarrow 1$ to num_clusters($som2\_out$) do
    $obs \leftarrow som\_out2[i].observations$;
    $num\_positives \leftarrow \sum_{j \in obs} y_2[j]$;
    $num\_negatives \leftarrow$ length($obs$) $- num\_positives$;
    if $num\_positives > 0$ then
        $k \leftarrow k + 1$;
        $clusters[k] \leftarrow i$;
        $values[k] \leftarrow num\_positives$;
        $weights[k] \leftarrow num\_negatives$;
    end
end
// Calculates the acceptable no.  of false positives
$weight_{max} \leftarrow \lfloor FP_{max} \times (n - \sum_i y[i]) \rfloor$;
$selected \leftarrow$ knapsack01($values, weights, weight_{max}$);
$p\_clusters2 \leftarrow clusters[selected]$;
```

Algorithm 2: Steps 1–3 of phase 2 (Classification with Constraint on False Positives)

the other ones we labeled as legitimate. More information about this dataset is available in previous works [7, 15].

We split the dataset described in a training and a test sets by random sampling with the following distributions:

- Training: 326 fraudulent listings and 21,422 legitimate ones;
- Test: 113 fraudulent listings and 21,914 legitimate ones.

Since many sellers (including fraudsters) post multiple listings, we took care that the listings of each seller appeared either in the training or in test set, so as to not artificially improve results.

5.1.2. Forest cover dataset. Some previous works on classification of imbalanced data [21, 24] used the Forest cover dataset [25] to test their proposals. This dataset comprises 581,012 observations of 54 cartographic variables. Each observation can belong to one of seven different forest cover types. The challenge is to predict the forest cover type of a region based on the cartographic variables. We followed the approach of Yuan and Ma [24]: we took as positive the observations

```
Input: A $n \times p$ matrix $\mathbf{X}$ with new data
Input: The trained maps $som1$ and $som2$
Input: The lists of positives clusters $p\_clusters1$ and $p\_clusters2$
Output: A vector $y$ of length $n$ with labels (0 or 1)
// Phase 1, step 4:  cluster new data
$som1\_out \leftarrow \text{apply\_som}(som1, \mathbf{X})$;
// Phase 1, step 5:  filter out negatives
for $i \leftarrow 1$ to $\text{num\_clusters}(som1\_out)$ do
    if $i \notin p\_clusters1$ then
        $y[som1\_out[i].observations] \leftarrow 0$;
    end
end
// Separate positives for phase 2
$\mathbf{X_2} \leftarrow \mathbf{X}[som1\_out[p\_clusters1].observations, :]$;
// Phase 2, step 4:  cluster new data
$som2\_out \leftarrow \text{apply\_som}(som2, \mathbf{X_2})$;
// Phase 2, step 5:  final classification
for $i \leftarrow 1$ to $\text{num\_clusters}(som2\_out)$ do
    if $i \in p\_clusters2$ then
        $y[som2\_out[i].observations] \leftarrow 1$;
    else
        $y[som2\_out[i].observations] \leftarrow 0$;
    end
end
```

Algorithm 3: Applying the trained maps to new data (steps 4 and 5 of both phases)

from classes 4 to 7, and as negative the remaining ones (classes 1 to 3), in order to make a dataset for binary classification, since our technique currently only works straightforward for this case. We chose 5% of the observations to make our training/test sets. The resulting dataset used for evaluation contained 91% of negative observations (majority class) and 9% of positive observations (minority class).

5.2. Parameter selection. The main parameters of our system were the sizes of the SOMs. Since the first SOM has a filtering role, what matters is to obtain the best possible false positives with a true positives rate close to 1. This will happen when the size of the SOM is not too big, not too small. A SOM that is too small will end up with positive observations in all clusters, rendering the filtering phase useless, since all observations will be sent to the next phase. Conversely, a SOM that is too big will lead to *overfitting*: there will be many clusters with very few or even just one positive observation. In practice this means that new observations will be labeled as positive only if they are very close to the positive observations present in the training set, what reduces the generalization power of the network. The overfitting effect is illustrated in Figure 5.1.

The size of the second SOM should be chosen so as to maximize the true positives for the given maximum false positives rate. Growing the SOM gives the knapsack

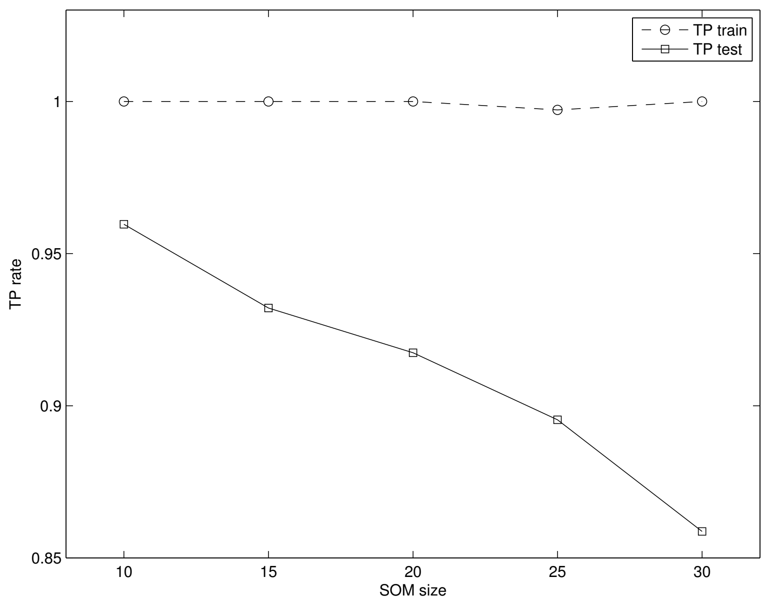


FIGURE 5.1. True positives for phase 1 versus SOM size (ML dataset)

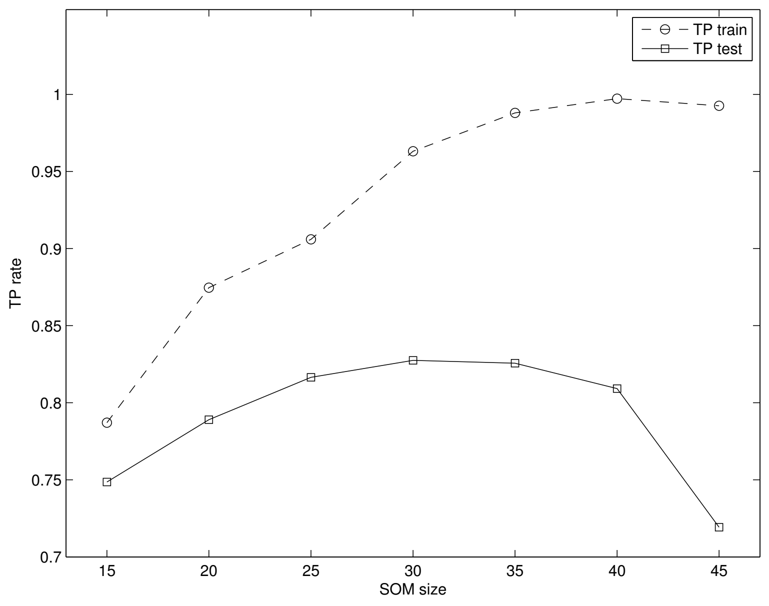


FIGURE 5.2. True positives for phase 2 versus SOM size (ML dataset)

algorithm more optimization opportunities, but also brings the problem of overfitting. This means that there is some map size that maximizes the true positives. This tradeoff is portrayed in Figure 5.2.

The strategy we adopted to find the optimal parameter set was to use 5-fold cross-validation on the training set, testing several combinations of map sizes. However, differently from what we did in our previous work [3], we used smaller maps for the first SOM and bigger maps for the second, since this yields better results.

5.3. Performance considerations. SOMs' training time grows linearly on the number of units and on the number of training examples. Since we use square maps, the number of cells grows quadratically on the map size. The number of training examples will usually be big due to class imbalance, so training time may be an issue. We should distinguish two moments: (i) parameter selection, (ii) training for effective application. Automatic parameter selection can take a long time, since there are many possible map size combinations to test. Some human guidance may help ruling out parameter combinations that are not likely to improve results. After maps sizes are chosen, the next step is to train the model with the whole training set. This can also take some time, but it needs to be done once, similarly to other machine learning methods. After some time it might be advisable to retrain the model, but a new parameter selection should not be necessary, unless data distribution changes significantly.

5.4. SOM configuration. Besides map size, there are other relevant design decision regarding the SOM map, namely topology and distance function. In our experiments we used the hexagonal topology, since we expected it to capture underlying distribution slightly better than grid topology. The distance function used was the *link distance*, which measures the minimum number of edges between the nodes in question. We did some exploratory testing varying topology and distance function; since the impact was only marginal, we stick to these options in order to avoid lengthy parameter selection procedures with very small gain.

We did not interfere with the learning rate of the SOM map, since the tool used took care of it automatically.

5.5. Results. In Table 1 we present the true and false positives values for different values of FP_{max}. These figures are averages over five runs of the algorithm, since SOM weights are initialized randomly, introducing some variability which can be alleviated through averaging several runs. In Table 2 we show the performance for each map separately for $FP_{max} = 20\%$.

To illustrate the results, we depict the SOMs for one run of the algorithm with $FP_{max} = 20\%$. We used the training data of the ML dataset. In Figure 5.3 we show the SOM for the filtering phase and in Figure 5.4 we display the SOM for the classification phase. To highlight the utility of SOMs to better understand data, one can see in the figures that fraudulent listings are *not* outliers; on the contrary, most of them are in regions with many listings.

Our results with the forest cover dataset are comparable to the existing ones: Yuan and Ma [24] obtained a geometric mean between true and false positives of 0.82 using a combination of SMOTE, genetic algorithms and Boosting, while we obtained 0.79 (for $FP_{max} = 20\%$) with a much simpler approach.

6. Conclusions

The main contribution of our work is a binary classification algorithm for highly imbalanced datasets. The algorithm is based on Self-Organizing Maps (SOMs) and cluster labeling based on the 0-1 knapsack algorithm, a novel combination which gives results easy to interpret using the visualization properties of SOMs. While other works already used SOMs for classification in the realm of fraud detection, they relied on the fact that fraudulent observations were outliers, a premise that is not valid in fraud prediction, which is the case for the ML dataset.

TABLE 1. Results of running the algorithm for different values of FP_{max}. Standard deviation is shown in parenthesis.

FP_{max}	ML		Forest cover	
	Average TPR	Average FPR	Average TPR	Average FPR
25%	82.7% (3%)	27.2% (1.5%)	85% (0.3%)	22% (0.3%)
20%	75.4% (4%)	22.4% (2.5%)	80.3% (0.7%)	17% (0.5%)
15%	65% (3%)	17.6% (2%)	73% (0.2%)	12.4% (0.2%)
10%	50.4% (3%)	11.1% (0.9%)	64.9% (0.9%)	8.1% (0.1%)
5%	29% (6%)	5.3% (0.2%)	48.7% (1.4%)	4.1% (0.2%)

TABLE 2. Performance per phase for $FP_{max} = 20\%$ (average of 5 runs)

	ML		Forest cover	
	TPR	FPR	TPR	FPR
Filtering phase	100%	91.5%	98%	75%
Classification phase	75.4%	25%	81.7%	23%

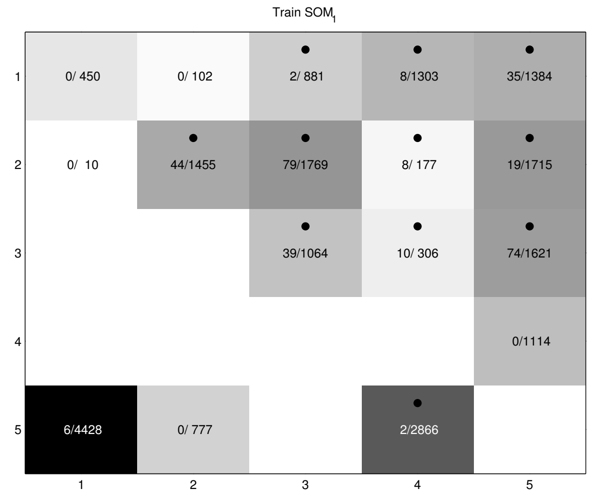


FIGURE 5.3. Training map for filtering phase. Figures refer to the number of positive observations in each cluster. All clusters with positive observations were labeled as positive, as shown by the dots.

Regarding the experimental results, the proposed algorithm predicted a substantial fraction of fraudulent listings in the ML dataset even with a highly degree of imbalanced (0.5% of fraudulent listings in the test set). The achieved performance was stable, since the standard deviation among different runs was small and the targeted false positives rate was respected. Nevertheless, false positives in the test set had a small bias, since they were consistently bigger than the desired FP_{max}. Results with the Forest cover dataset were similar, being comparable to existing results in the literature. A visual inspection of the generated maps for the ML dataset confirmed that fraudulent listings were not outliers, since they were usually mixed with many other legitimate listings.

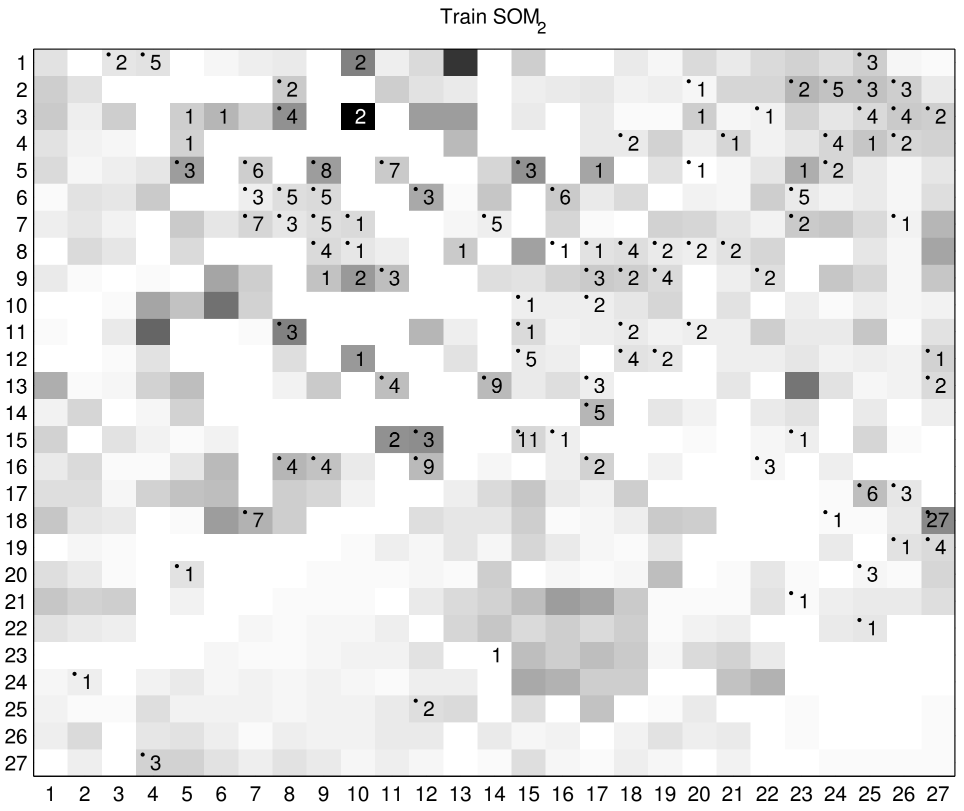


FIGURE 5.4. Training map for classification phase. The number of positive observations in each cluster is shown (if any). The total number of observations is represented by the background color (darker color means more observations). Clusters with the spot were labeled as positive by the knapsack algorithm.

Besides being resistant to the class imbalance problem, the proposed algorithm also has the advantage of simplicity and easiness of interpretation, since it reduces the prediction problem to labeling the appropriate SOM cells as positive, which is equivalent to choosing which regions of the feature space should be treated with more attention. Differently from other works, our algorithm relies neither on the values of the SOM's weight vectors nor on more complex calculations: it goes directly from clustering to labeling based solely on the counts of observations in each SOM unit.

The use of two SOMs also had its advantages: the first map reduced significantly the number of observations to be further analyzed, since it has a very high true positives with a moderate false positives. Another advantage of the proposed algorithm is that it automatically chooses the SOMs' sizes through cross-validation on the training set, freeing the user from specifying these parameters.

As future work we want to rank clusters, in order to give the user the chance to concentrate his efforts in the most relevant observations. We also would like to test with other imbalanced datasets to see if the same results hold.

Acknowledgments

This work was sponsored by University of Bucharest, under postdoctoral research grant nr. 15515 of September 28th, 2012.

References

[1] Bezalel Gavish and Christopher Tucci. Reducing internet auction fraud. *Communications of the ACM*, 51 (5), 2008.

[2] Dawn G. Gregg and Judy E. Scott. A typology of complaints about ebay sellers. *Communications of the ACM*, 51 (4):69–74, 2008.

[3] Vinicius Almendra and Denis Enachescu. Using self-organizing maps for fraud prediction at online auction sites. In *Proceedings of the 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2013)*, Timisoara, Romania, 2013. IEEE Computer Society.

[4] R.J. Bolton and D.J. Hand. Statistical fraud detection: A review. *Statistical Science*, 17(3):235–255, 2002.

[5] Bezalel Gavish and Christopher Tucci. Fraudulent auctions on the internet. *Electronic Commerce Research*, 6(2):127–140, April 2006.

[6] Dawn G. Gregg and Judy E. Scott. The role of reputation systems in reducing on-line auction fraud. *International Journal of Electronic Commerce*, 10(3):95–120, 2006.

[7] Vinicius Almendra. A comprehensive analysis of nondelivery fraud at a major online auction site. *Journal of Internet Commerce*, 11(4):309–328, 2012.

[8] Wen-Hsi Chang and Jau-Shien Chang. A novel two-stage phased modeling framework for early fraud detection in online auctions. *Expert Systems with Applications*, 38(9):11244–11260, September 2011.

[9] Duen Horng Chau and Christos Faloutsos. Fraud detection in electronic auction. In *Proceedings of European Web Mining Forum*, 2005.

[10] Chaochang Chiu, Yungchang Ku, Ting Lie, and Yuchi Chen. Internet auction fraud detection using social network analysis and classification tree approaches. *International Journal of Electronic Commerce*, 15(3):123–147, April 2011.

[11] Rafael Maranzato, Adriano Pereira, Alair Pereira do Lago, and Marden Neubert. Fraud detection in reputation systems in e-markets using logistic regression. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1454–1455, Sierre, Switzerland, 2010. ACM.

[12] Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. NetProbe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, WWW 2007, Banff, Alberta, Canada, 2007. ACM Press.

[13] Liang Zhang, Jie Yang, Wei Chu, and Belle Tseng. A machine-learned proactive moderation system for auction fraud detection. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, page 2501–2504, New York, NY, USA, 2011. ACM.

[14] Liang Zhang, Jie Yang, and Belle Tseng. Online modeling of proactive moderation system for auction fraud detection. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, page 669–678, New York, NY, USA, 2012. ACM.

[15] V. Almendra. Finding the needle: A risk-based ranking of product listings at online auction sites for non-delivery fraud prediction. *Expert Systems with Applications*, 40(12):4805–4811, September 2013.

[16] Shi-Jen Lin, Yi-Ying Jheng, and Cheng-Hsien Yu. Combining ranking concept and social network analysis to detect collusive groups in online auctions. *Expert Systems with Applications*, 39(10):9079–9086, August 2012.

[17] Andrei Sorin Sabau. Survey of clustering based financial fraud detection research. *Informatica Economica Journal*, 16(1):110–122, 2012.

[18] Vladimir Zaslavsky and Anna Strizhak. Credit card fraud detection using self-organizing maps. *Information and Security*, 18:48, 2006.

[19] J.T.S. Quah and M. Sriganesh. Real time credit card fraud detection using computational intelligence. In *International Joint Conference on Neural Networks, 2007. IJCNN 2007*, pages 863–868, 2007.

[20] Vinicius Almendra and Bianca Roman. Using exploratory data analysis for fraud elicitation through supervised learning. In *13th International Symposium on Symbolic and Numeric*

Algorithms for Scientific Computing (SYNASC 2011), pages 251–254, Timisoara, Romania, 2011.

[21] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[22] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[23] Vinicius Almendra and Denis Enachescu. A fraudster in a haystack: Crafting a classifier for non-delivery fraud prediction at online auction sites. In *Proceedings of the 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2012)*, Timisoara, Romania, 2012. IEEE Computer Society.

[24] Bo Yuan and Xiaoli Ma. Sampling + reweighting: Boosting the performance of AdaBoost on imbalanced datasets. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, 2012.

[25] Jock Blackard. Covertype data set, 1998.

Faculty of Mathematics and Informatics University of Bucharest Strada Academiei 14, Bucharest - sector 1 010014 Romania

E-mail: vinicius.almendra@gmail.com

Faculty of Mathematics and Informatics University of Bucharest Strada Academiei 14, Bucharest - sector 1, 010014, Romania

"Gheorghe Mihoc - Caius Iacob" Institute of Mathematical Statistics and Applied Mathematics of Romanian Academy Calea 13 Septembrie 13, Bucharest - sector 5, 050711, Romania

E-mail: denaches@fmi.unibuc.ro