A Supervised Auto-Tuning Approach for a Banking Fraud Detection System

Michele Carminati, Luca Valentini, and Stefano Zanero

Politecnico di Milano, Italy
Dipartimento di Elettronica, Informazione e Bioingegneria
{michele.carminati,stefano.zanero}@polimi.it
luca.valentini@mail.polimi.it

Abstract In this paper, we propose an extension to Banksealer, one of the most recent and effective banking fraud detection systems. In particular, until now Banksealer was unable to exploit analyst feedback to self-tune and improve its performance. It also depended on a complex set of parameters that had to be tuned by hand before operations.
To overcome both these limitations, we propose a supervised evolutionary wrapper approach, that considers analyst's feedbacks on fraudulent transactions to automatically tune feature weighting and improve Banksealer's detection performance. We do so by means of a multi-objective genetic algorithm.
We deployed our solution in a real-world setting of a large national banking group and conducted an in-depth experimental evaluation. We show that the proposed system was able to detect sophisticated frauds, improving Banksealer's performance of up to 35% in some cases.

Keywords: Internet banking; Fraud detection; Genetic algorithm; Supervised learning

1 Introduction

Nowadays, Internet banking has become one of the major target of fraudulent cyber-attacks such as as phishing, malware, and trojan infections, and has brought to a worldwide loss of billions of dollars every year [36,4]. According to Kaspersky, in 2016 financial malware infected about 2,8 millions personal devices, a 40% increase since 2015 [1].

To contrast fraudulent cyber-attacks, banks developed fraud analysis and detection systems that aim at identifying unauthorized activities as quickly as possible. These systems monitor and scrutinize transactions, scoring suspicious ones for analyst verification. In spite of the importance of the subject, very little research is openly carried out, because of privacy restrictions and difficulties in obtaining real-world data.

Thanks to the collaboration with a major banking group, we were able to develop Banksealer [7,8], a novel, unsupervised fraud analysis system that automatically ranks frauds and anomalies in banking transactions. While the experiments presented in [8] showed that Banksealer is an effective approach in

identifying frauds and seems to provide a meaningful support to the banking analysts in fraud investigation, one of its main limitations is the inability to collect and exploit banking analyst's feedback. It also depended on a complex set of parameters that had to be tuned by hand before operations.

To overcome these limitations, in this paper we propose a general supervised evolutionary wrapper approach that considers analyst feedback on fraudulent transactions to find an optimal tuning of Banksealer's parameters. Our approach implements the Non-dominated Sorting Genetic Algorithm II (NSGA-II) to find a configuration of parameters that optimizes the ranking of potential frauds at runtime. Most of the parameters are feature weights, which makes the task particularly challenging, since we are confronted with large and unbalanced datasets in which there are multiple variants of frauds that are, overall, extremely rare (i.e. less than 1% with respect to legitimate transactions), and dynamically evolve over the time.

We deployed our solution in a real-world setting of a large national banking group and conducted an in-depth experimental evaluation. Thanks to collaboration with this bank and leveraging the domain expert's knowledge, we reproduced frauds (in a controlled environment) performed against online banking users, and recorded the resulting fraudulent transactions. We show that the proposed system was able to detect sophisticated frauds improving Banksealer's performance up to a factor of 35%.

In summary, in this paper we make the following novel contributions:

- We propose a supervised learning module based on Multi-Objective-Genetic-Algorithm (MOGA) able to automatize the feature weighting task and to improve detection performances of Banksealer.
- We free banking analysts from the manual job of the feature weighting task and exploit their knowledge analyzing their feedback.
- We improve Banksealer's ability to evolve over the time and to adapt itself to changes in both threats and user behavior.

2 Overview of Banksealer and Goals

In this section we will recall the main concepts underlying the existing Banksealer system, insofar as they are needed to explain the motivation of the present work. We refer the interested reader to the original paper [8] for additional details.

Banksealer characterizes the customers of the bank by means of a local, a global, and a temporal *profile*, which are built during a training phase taking as input a list of *transactions*. Each type of profile extracts different statistical *features* from the transaction attributes, according to the type of model built. A list of the employed attributes is presented in Table 1.

Once the profiles are built, Banksealer processes new transactions and ranks them according to their anomaly score and the predicted risk of fraud. The *anomaly score* quantifies the statistical likelihood of a transaction being a fraud w.r.t. the learned profiles. The *risk of fraud* prioritizes the transactions, combining the anomaly score with the transaction amount. Banksealer provides the

analysts with a ranked list of potentially fraudulent transactions, along with their anomaly score.

The *local* profile characterizes each user's individual spending patterns. During training, we aggregate the transactions by user and compute the empirical marginal distribution of the features of each user's transactions (for simplicity, we do not consider correlation between features). This representation is simple and effective, and hence is indeed directly readable by analysts who get a clear idea of the typical behavior by simply looking at the profile. At runtime, we calculate the anomaly score of each new transaction using a modified version of the Histogram Based Outlier Score (HBOS) [14] method. HBOS computes the log-likelihood of a transaction according to the marginal distribution learned. The HBOS score is a weighted sum:

$$HBOS(t) = \sum_{0<i\leq d} w_i * \log \frac{1}{f(t_i)}; \qquad \sum_{0<i\leq d} w_i = 1$$

where w_i is the weighting coefficient of the i-th feature, that allows analysts to tune the system. It is worth noting that $f(t_i)$ is the application of the min-max normalization [15, pp. 71–72] to the frequency hist_i of the i-th feature. This normalization was necessary in order to account the variance of each feature.

The *global* profile characterizes "classes" of spending patterns, clustering users together. Each user is represented as a feature vector of six components: total number of transactions, average transaction amount, total amount, average time span between subsequent transactions, number of transactions executed from foreign countries, number of transactions to foreign recipients (bank transfers dataset only). To find classes of users with similar spending patterns, we apply an iterative version of the Density-Based Spatial Clustering of Applications with Noise (DBSCAN), using the Mahalanobis distance [22] between the aforementioned vectors. We assign to each user global profile an anomaly score, which tells the analyst how "uncommon" the spending pattern is with respect to other customers. For this, we compute the unweighted-Cluster-Based Local Outlier Factor (CBLOF) [3] score, which considers small clusters as outliers with respect to large clusters. More precisely, the more a user profile deviates from the dense cluster of "normal" users, the higher his or her anomaly score will be.

The global profile is also leveraged to mitigate the issue of *undertraining*. Undertrained users are users that performed a low number of transactions, and represent a relevant portion of a typical dataset. For undertrained users, we consider their global profile and select a cluster of similar users.

Table 1. Attributes for each type of transaction. Attributes in **bold** are hashed for anonymity needs.

Dataset	Attributes
Bank Transfers	Amount, CC_ASN, **IP**, **IBAN**, IBAN_CC, Timestamp
Phone recharges	Amount, CC_ASN, **IP**, Phone operator, **Phone number**, Timestamp
Prepaid Cards	Amount, Card type, **Card number**, CC_ASN, **IP**, Timestamp

Finally, the *temporal* profile deals with frauds that exploit the repetition of legitimate-looking transactions over time, by comparing the current spending profile of the user against their history. During training, we extract the mean and standard deviation of the following aggregated features for each user: total amount, total and maximum daily number of transactions. At runtime, according to the sampling frequency, we calculate the cumulative value for each of the aforementioned features for each user, and compare it against the previously computed metrics.

2.1 Research goal

While the experiments presented in [8] showed that Banksealer is an effective approach in identifying frauds and seems to provide a meaningful support to the banking analysts in fraud investigation, one of its main limitations is the lack of a bi-directional communication channel between the unsupervised system and the banking analyst to collect and exploit analyst's feedback and knowledge to improve detection performance. Furthermore, Banksealer works with an empirical configurations of weights, manually set by analysts.

Therefore, the focus of this work is to overcome this limitation by exploiting banking analysts' feedback to auto-tune Banksealer. With auto-tune we mean to find an optimal features weights configuration used by Banksealer to compute the transaction's anomaly score.

This is basically an instance of the *feature weighting* problem, a variant of the more common *feature selection* one [38]. The difference is that in feature selection we basically assign a binary weight to discard redundant or irrelevant attributes. Feature weighting instead assigns real-valued weights to each feature, based on relevance [37,10,33].

Feature weighting algorithms can be classified into two categories, based on whether or not the feature weighting is done independently from the detection learning algorithm. If feature weighting is done independently from the detection task, the technique is said to follow a *filter approach*. Otherwise, it is said to follow the *wrapper approach*. A *filter approach* works exclusively on the data and uses probabilistic dependence measures to determine correlations among features. Being independent of the detection task means that changes to the detection system do not impact assigned weights. Also, it is more efficient than a wrapper approach from the computational point of view, since the detection system does not need to be ran to evaluate the candidate weights configurations.

On the other hand, a *wrapper approach* [19] consists of executing feature weighting, intertwined with the detection task. Wrapper approaches usually outperform *filter approaches* from a detection point of view, but obviously they are computationally more demanding.

In this work we opted for the more computationally intensive *wrapper approaches*. In particular, we made use of *genetic algorithms*, as they can handle multiple local optima and are designed to support also multiple objective criteria [38]. In fact, as we will detail in the following, our problem requires to trade off between multiple objectives. For a deeper and more exhaustive notion

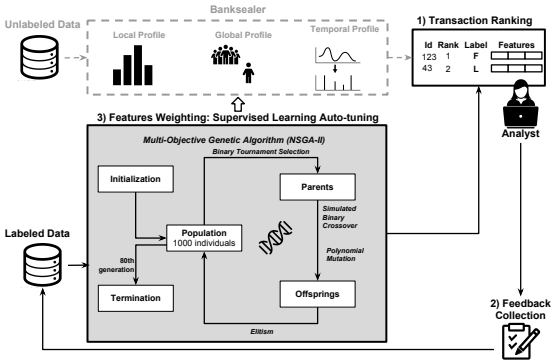

Figure 1. Logical view of Banksealer integrated with the feature weighting module.

of Genetic Algorithm (GA) we invite the reader to refer to [25,13,24] as this is beyond the scope of this paper.

3 Approach Overview

Our approach to solve the feature weighting problem stated in § 2.1 is summarized in Fig. 1. It is composed of three logical steps:

1. **Transaction Ranking**. Banksealer generates in output a ranking of the transactions based on the fraud risk score. As shown in § 2, Banksealer uses the HBOS score to compute the anomaly score of a transaction by combining the weighted contribution of each feature. The formula can be simplified as follow $HBOS(t) = \sum_{0<i\leq d} w_i \cdot c_i$, where d is the number of features of the dataset, w_i and c_i are respectively the weight and the score contribution of the i^{th} feature.
2. **Feedback Collection**. Analysts, after going through the ordered transaction list, flag as *fraud* transactions that have been verified to be fraudulent, or as *Suspect* the ones that – even if they turned out to be benign – were definitely anomalous enough to warrant investigation. All other transactions are benign. The labeled dataset is the input for the feature weighting process.
3. **Feature Weighting**. After having collected all transaction feedbacks in a labeled dataset, our solution follows the wrapper approach. We opted to use a GA, and specifically Non-dominated Sorting Genetic Algorithm II (NSGA-II) [12]. The basic idea we follow is to generate a population that represents different feature weight configurations, and then evaluate the accuracy of the fraud detection system for each candidate, by calling the Banksealer testing function on each individual of the population. We will describe in detail the fitness function used for evaluation, operators, and other details of the application of the algorithm in § 4.3.

While describing the specificities of Non-dominated Sorting Genetic Algorithm II (NSGA-II) is beyond the scope of this paper, and we refer the reader to the original work [12], it is relevant to point out that the algorithm exhibits a time complexity of $O(MN^2)$ and a spatial complexity of $O(N^2)$ (where M is the number of objectives and N is the population size). It also implements elitism that can be shown [39,34,27,29] to speed up the performance of GAs significantly. Finally, NSGA-II is parameterless regarding the sharing mechanism used to introduce diversity in the population, which suits our purpose of making the tuning mechanism completely automated.

4 Approach Implementation

In this section we describe in detail the application of the NSGA-II to our problem. We describe the encoding scheme, the handling of constraints, the selection of operators, and the fitness function used.

4.1 Encoding Scheme and Constraints

The first step in designing a GA is the representation of the genes in an individual (i.e., encoding scheme). For our feature weighting problem, this is straightforward. In fact, we decide to implement the weight configuration as a list of real numbers. Each cell of the list is associated to a feature of those used by Banksealer (see Table 1), and the value that each cell contains represents the weight associated to the feature.

Furthermore, at this stage we must consider also possible constraints that may influence our encoding scheme design, but above all the implementation of crossover and mutation operators that we will present later. Our problem has two constraints:

1. $w_i \in [0, 1]$, which means that the weight w_i of the i^{th} feature must be a real value between 0 and 1.
2. $\sum_{i=1}^{F} w_i = 1$, which means that the sum of all weights belonging to a configuration must be equal to 1 (F is the total number of features).

Both constraints can be satisfied by normalizing to 1 the sum of all genes values whenever a new individual is created, and by ensuring that each values is positive and real.

4.2 Population

Another very important aspect of GA is the number of individuals in the population. A small population size leads to a faster convergence of the fitness functions. However, the drawback is that this might get the algorithm stuck in local optima.

Therefore, we had to find a trade-off for the population size to get a good solution in a reasonable time. In Fig. 2, we report the evolution of fitness functions according to population size. In all three cases, we get similar results. We can see that for smaller population sizes the algorithm stagnates in local optima for several generations. For population sizes of 500 and 1000, we have almost the same performance, and no relevant improvements are obtained by increasing population to 5000. As a consequence, we chose a population size of 1000 since it represents the optimal trade-off between diversity and performance.

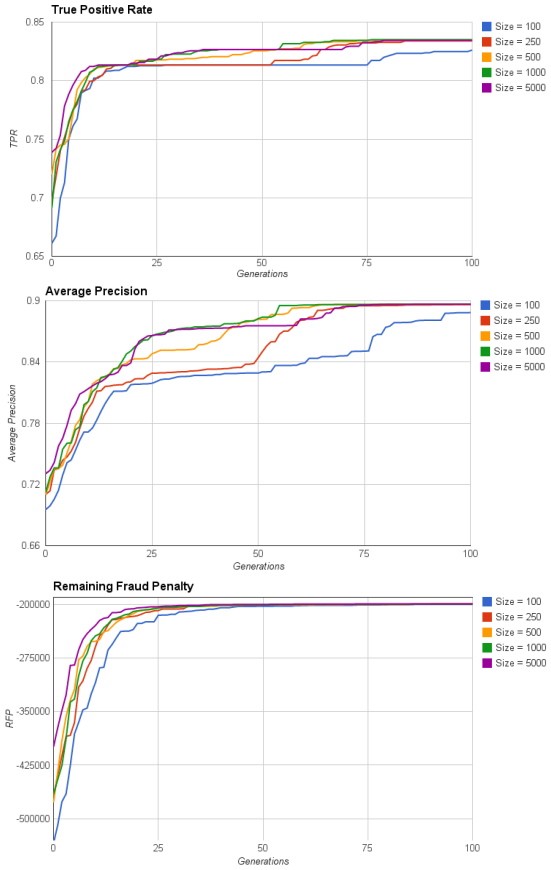

Figure 2. Population size estimation.

4.3 Fitness Functions

Fitness functions are needed to evaluate feature weights configurations and allow to choose the best ones. We choose three fitness functions: *True Positive Rate*,

Average Precision, and *Remaining Frauds Penalty*. The first two are defined in § 5.4, and they are both meant to be maximized. The *Remaining Frauds Penalty* (RFP) assigns a penalty (equal to the number of frauds not yet detected) for each normal transaction detected as fraudulent: RFP $= \sum_{k=1}^{R} RF(k) \times N(k)$, where R is the number of total transactions in the ranking and $RF(k)$ is the number of frauds not yet detected at the k^{th} position of the ranking; $N(k) = 1$ if the k^{th} transaction is normal, 0 otherwise. In this case, we want to minimize Remaining Frauds Penalty (RFP).

The reason for choosing TPR is self-explaining: our main goal is to detect as many as possible frauds in the top N positions of the ranking. Its main drawback, however, is that it does not keep into consideration how frauds are arranged in the ranking: it neither considers how "high" are frauds present in the top N positions, nor how "far" down are the missed frauds. To overcome this problem, we use AP and RFP in addition to TPR. By doing this, we reduce the spread of frauds and push them from the lower positions to the upper part of the ranking. In particular, minimizing the RFP value, we push up complex frauds from the very last positions of the ranking. However, RFP tends to penalize more frauds at the bottom of the ranking. Instead, maximizing the Average Precision (AP), we gather frauds together reducing the distance between sequent fraudulent transactions. However, it has less influence on frauds in the last positions.

As a consequence, the combination of these three fitness functions makes the algorithm stable also in very complex scenarios.

4.4 Operators

The choice of selection, crossover and mutation operators is domain specific. Furthermore, we must find a good trade-off between *exploitation* (using knowledge already available to find better solutions) and *exploration* (investigating new and unknown areas in the search space).

For the *selection* operator we picked a *tournament selection*: the operator selects K random individuals from the population, and compares them. The winner is inserted into the mating pool. The tournament repeat until the mating pool is filled. The mating pool, being comprised of tournament winners, has a higher average fitness than the average population fitness. In particular, we used a *binary tournament* selection, i.e. $K = 2$. We prefer this operator because its selection pressure is not high, and it assures genetic diversity, reducing the risk of converging to local optima. On the other hand, it does not take a lot of time to converge thanks to the elitist property guaranteed by NSGA-II. It is also proven to be robust in the presence of noise [23].

For the crossover operator, we chose the *simulated binary crossover* [11]. The operator computes offspring solutions $x_i^{(1,t+1)}$ and $x_i^{(2,t+1)}$ from parent solutions $x_i^{(1,t)}$ and $x_i^{(2,t)}$ by defining the spread factor β_i as the ratio of the absolute difference in offspring values to that of the parent values:

$$\beta_i = \left| \frac{x_i^{2,t+1} - x_i^{1,t+1}}{x_i^{2,t} - x_i^{1,t}} \right|$$

The spread factor β_i is distributed according to the following probability distribution:

$$P(\beta_i) \begin{cases} 0.5(\eta_c+1)\beta_i^{\eta} & \beta \leq 1 \\ 0.5(\eta_c+1)\frac{1}{\beta_i^{\eta+2}} & \text{otherwise} \end{cases}$$

Where η_c is a parameter we set to control the variance of the distribution: A large value of η_c gives a higher probability to create offspring "near" the parents, while a small one allows distant solutions to be selected as offspring. For our problem we set the probability of crossover between two parents to 0.9, and $\eta_c = 5$. This because in our tests greater values of η_c resulted in a slow down of the algorithm convergence, since creating offspring very similar to their parent favors exploitation much more than exploration.

We decide to use the *simulated binary crossover* for three reasons:

- It is designed for offspring with real variables, like our weights.
- It preserves parents schemata in the offspring. By doing this, the crossover operator does not destroy every time the solution creating a new one very different from the parents.
- It has a very interesting self-adaptation property: the location of the offspring solution depends on the difference in parent solutions. If the difference in the parent solution is small, the difference between the offspring and parent solutions is also small and vice-versa.

Finally, we chose the *polynomial mutation* as mutation operator. It attempts to simulate the offspring distribution of binary-encoded bit-flip mutation on real-valued variables. This operator is usually used in pair with simulated binary crossover because it works in a very similar way, favoring mutated offspring nearer to the parents. Adopting the same notation used before for crossover, a new mutated offspring is obtained as $x_i^{t+1} = x_i^t + (x_i^U - x_i^L)\delta_i$, where x_i^U and x_i^L are respectively the upper bound and the lower bound of the variable at position i. Instead, δ_i is defined as:

$$\delta_i = \begin{cases} (2r_i)^{\frac{1}{\eta_m+1}} - 1 & r_i \leq 0.5 \\ 1 - [2(1-r_i)]^{\frac{1}{\eta_m+1}} & \text{otherwise} \end{cases}$$

where, similarly to crossover, r_i is a random number between 1 and 0, and η_m is the mutation distribution index. We set $\eta_m = 10$, in such a way that resulting offspring are different but rather close to the non-mutated individual. We choose this high mutation rate, because it allows to obtain a high diversity on the Pareto front. Instead, the probability of mutating a single variable is equal to $\frac{1}{F}$ where F is the number of total features used by the algorithm. This results in one mutation per offspring on average (which corresponds roughly to the idea of "shifting the value of one variable").

5 Experimental Evaluation

In this section we describe the experimental evaluation of our learning module integrated with Banksealer.

	Type of Fraud	IP Country	IBAN country
Scenario 1	Information Stealing	foreign	foreign
Scenario 2	Information Stealing	foreign	Italian
Scenario 3	Information Stealing	Italian	foreign
Scenario 4	Information Stealing	Italian	Italian
Scenario 5	Transaction Hijacking	-	foreign
Scenario 6	Transaction Hijacking	-	Italian

Table 2. Scenarios of fraudulent activities.

5.1 Hardware and computation times

Our experiments have been executed on a desktop computer with the following specifications: Quad-core 3.40 GHz Intel i7-4770 CPU, 16GB of RAM, and the Linux kernel 3.7.10 x86_64. The results of the experiments we made are obtained computing the average on 30 tests to avoid statistical oscillations. In average, a single weighting process of 80 generations on a one-month dataset lasts 1 hour and 30 minutes. We obtained this execution time thanks to the parallel fitness function evaluations that resulted to be about 3 times faster than the non-parallel version.

5.2 Dataset

The dataset in our possession belongs to an important Italian banking institute and is anonymized to protect privacy of customers. The dataset covers the period from April 2013 to August 2013. We split the data in a *training dataset*, used to train Banksealer, consisting of 3 months of data; a *weighting dataset* of one month, containing the analyst feedback and used to learn the optimal configuration of weights; and finally, a *testing dataset*, consisting of the last month of data (and also containing analyst feedback). We show the results on the bank transfer data for brevity, but similar results can be obtained for the other contexts such as prepaid cards and phone recharges.

5.3 Synthetic Fraud Scenarios

The dataset under analysis does not contain frauds. Therefore, as already successfully done in [7,8], we inject fraud scenarios that replicate the typical attacks performed against online banking users. We consider two types of fraudulent attacks, reconstructed using domain expert and analyst advice:

– **Information stealing scenario**. It simulates a banking trojan or a phishing attack in which the customer is deceived into entering its credentials and a one-time-password (OTP). The stolen informations are sent to the fraudster, who uses them to execute a transaction towards an unknown IBAN. In this scenario, we suppose that the fraudster is interested into stealing as much money as he or she can. As a consequence the amount transferred will be very high, from 10.000€ to 50.000€. The fraudster can connect to the bank server from an Italian or a foreign IP address and money can be transfered to

an Italian or foreign IBAN. To inject the transactions, we randomly choose a user between those present in the testing dataset and we inject a transaction with a random timestamp.
– **Transaction hijacking scenario**. It simulates the infection of the user's computer by a MitB attack. The malware deceive the user into entering two OTPs and then exploit its capabilities to execute a second transaction using the user's browser. In this scenario, the transaction is still directed toward an unknown IBAN, which can be Italian or foreign, however the connection is executed from the victim's computer. As in the Information stealing scenario, we suppose that the fraudster is interested into stealing as much money as possible and, as a consequence, the amount transferred will be very high, from 10.000€ to 50.000€. To inject the transaction we randomly choose a user between those present in the testing dataset and after that we randomly select one of his or her transaction. The injected transaction will be executed no more than ten minutes after the selected transaction to simulate the MitB session hijacking.

In Table 2 we report the synthetic scenarios and their characteristics.

5.4 Metrics

Given the nature of our system, that "ranks" transactions according to anomaly, we need to slightly redefine the traditional evaluation metrics. We define as "positive" any transaction scored among the top N positions of the ranking, where N is the number of fraudulent transactions in the dataset. In our experiment we inject synthetic fraudulent transactions equivalent to the 1% of the dataset.

A True Positive (TP) is a fraudulent transaction that appears in the first N positions of the ranking. We similarly define True Negative (TN), False Positive (FP), and False Negative (FN). Then we use the traditional definition of *True Positive Rate (TPR)*:

$$TPR = \frac{TP}{TP + FP}$$

We also compute the *Average Precision (AP)*, which takes into account the position of fraudulent transactions in the ranking:

$$\text{AP} = \frac{\sum_{k=1}^{R} P(k) \times F(k)}{N}$$

where R is the number of total transactions in the ranking, $P(k) = \frac{\#frauds}{k}$, and $F(k) = 1$ if the k^{th} transaction is fraudulent, 0 otherwise.

Since our dataset is highly unbalanced in favor of normal transactions, we use the Matthews Correlation Coefficient (MCC) and Average Accuracy (AA) metrics, because they are less affected by this problem.

The MCC expresses the relationship between the observed and predicted binary classifications ($MCC = 1$ perfect prediction, $MCC = 0$ no better than random prediction, and $MCC = -1$ total disagreement between prediction and observation):

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

The AA is an average of the accuracy obtained for both fraudulent and normal transactions classes:

$$AA = \frac{1}{2}\left[\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right]$$

5.5 Experiment 1

In this experiment, we want to verify the quality of the *Weighted Banksealer* on single scenarios listed in Table 2, and compare it with *Banksealer*. We use the dataset described in § 5.2. We compute the TPR of both systems as defined in § 5.4. The results of our experiment can be seen in Table 3.

It is evident that feature weighting brings an improvement in almost all scenarios under analysis. Since *Banksealer* already guarantees good fraud detection performance in the information stealing scenario, the most significant gains are in the hijacking scenario, which is the most complex to detect. In particular, when looking at the limited results for Scenario 6, keep in mind that the fraudulent transactions are almost indistinguishable from benign ones in this case.

5.6 Experiment 2

With the objective of verifying the quality of *Weighted Banksealer* on a scenario closer to the real world, we test our approach against a "mixed scenario" obtained injecting in the dataset the same number of frauds, but randomly extracted from all of the scenarios of the first experiment. The results can be seen in Table 4. As we can see, *Weighted Banksealer* gets a *TPR* higher than *Banksealer* with a difference of 23%. But we also improve the ranking, concentrating most frauds in the top positions. This is expressed by the Average Precision, or it can be seen in Figure 3, where we plot the cumulative distribution of the detection ordered by ranking. The yellow line models the detection performance

Fraud scenario	IP	IBAN	**BS**			**BSW**			Improvements		
			TPR (%)	AA (%)	MMC	TPR (%)	AA (%)	MMC	TPR (%)	AA (%)	MMC
1: Information Stealing	Foreign	Foreign	97	98	0.97	98	99	0.98	+1	+1	+0.01
2: Information Stealing	Foreign	National	91	95	0.91	94	97	0.94	+3	+2	0.03
3: Information Stealing	National	Foreign	97	98	0.97	97	98	0.97	0	+0	0
4: Information Stealing	National	National	91	95	0.91	92	96	0.92	+1	+1	+0.01
5 - Transaction Hijacking	-	Foreign	75	87	0.77	95	97	0.95	+20	+10	+0.18
6 - Transaction Hijacking	-	National	22	68	0.34	57	78	0.63	+35	+10	+0.29

Table 3. Experiment 1: TPR, AA and MCC results of Experiment 1. BS = Banksealer, BSW = Weighted Banksealer

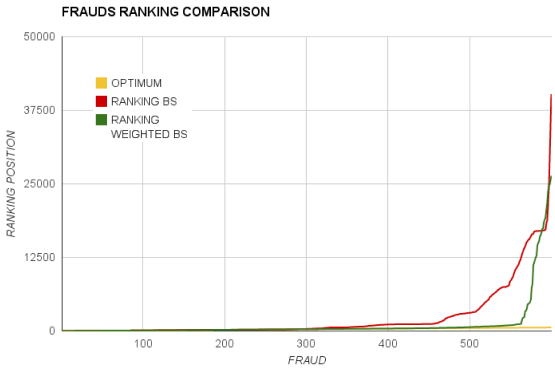

Figure 3. Ranking comparison for Experiment 2.

of an ideal fraud detection system. It is evident that *Banksealer* diverges earlier than *Weighted Banksealer* from it. For a further comparison of *Banksealer* and *Weighted Banksealer* we report in Figure 4 also the *Receiver Operating Characteristic* curve (*ROC*), which confirms the better overall performance of *Weighted Banksealer*.

5.7 Overfitting Problem

In the design of the system and during the experimental evaluation, we put great effort in limiting overfitting (a real issue in noisy, unbalanced datasets like ours).

In our problem, overfitting may be caused by an over-weighting of the system caused by the execution of too many generation of the GA. In that case, we could see that the performance over the weighting dataset keeps to increase, while performance decreases on validation dataset, which contains data unseen by the system. To limit overfitting we study how many generations are needed to learn the weights configuration and we stop the algorithm as soon as it starts to learn noise. This approach is usually called *early stopping*. We wait that all three functions converge and reach an equilibrium and as we can see this happens at 80^{th} generation. In fact, after 80^{th} generation in all three functions for several generations no relevant improvements are obtained. The performance on the validation dataset starts to get worse around the 115^{th} generation for the Average Precision fitness function. It is not very visible, but performance on weighting dataset are increasing. In addition, after some generations we can see

	TPR	Average Precision	Matthews Correlation Coefficient	Average Accuracy
Banksealer	58%	68%	0.67	82%
Weighted Banksealer	81%	88%	0.83	91%
Improvements	**+23%**	**+20%**	**+0.16**	**+9%**

Table 4. Results of Experiment 2.

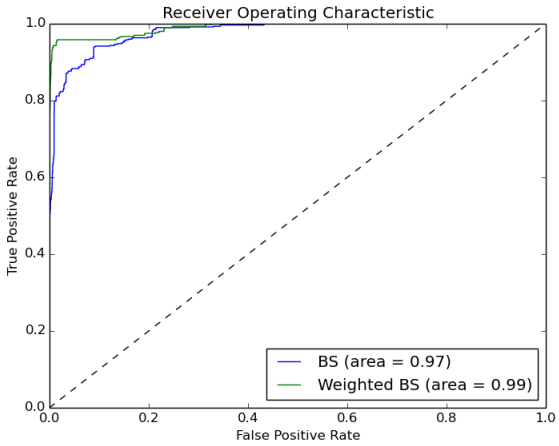

Figure 4. ROC curve for Experiment 2.

also that the other functions start to be affected by overfitting. In Figure 5 we report the results of our overfitting test.

In addition, we put effort to produce synthetic transactions that resemble the real ones to be as realistic as possible and to avoid the overfitting of our approach to "trivial" fraudulent transactions. To evaluate the quality of simulated data with respect to the real one, we empirically compare the distribution of transactions features by applying the kernel density estimation method [30] and box-plot diagrams. In addition, we applied the non- parametric two-sample permutation test for the comparison between central tendency of the features. With respect to other non-parametric tests it does not require verification of any assumption about distribution's shape and variability of the two samples. The null hypothesis specifies that the permutations are all equally likely with a significance level $\alpha = 0.05$. In other words, the distribution of the data under the null hypothesis satisfies exchange-ability. Since we found a $p - values > \alpha$, we failed to reject the null hypothesis that the samples are drawn from the same distribution.

6 Related Works

Fraud detection, mainly focused on credit card fraud, is a wide research topic, for which we refer the reader to [9,31,5]. In this section we focus on the feature-weighting task.

The *filter approach* has been used in [26] where is presented an unsupervised feature selection algorithm suitable for dataset containing a lot of dimension. The method is based on measuring similarity among features using the maximum information compression index. [32] is an example of application of this algorithm in bioinformatics field. *FOCUS* [2] is a feature selection algorithm for noise-free Boolean domains. It exhaustively examines all subsets of features, selecting the

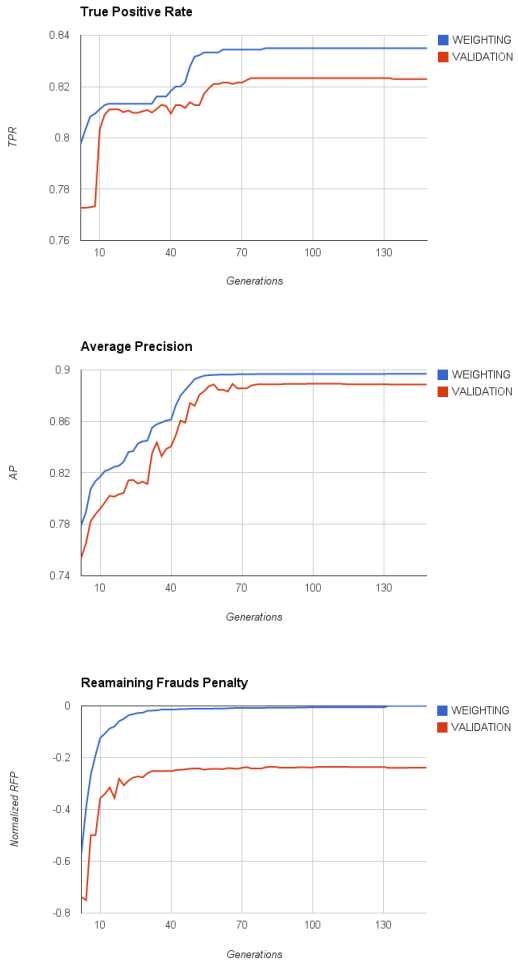

Figure 5. Overfitting analysis on the different fitness functions.

minimal subset of features sufficient to determine the label value for all instances in the training set. The relief algorithm [20,17] assigns a weight to each feature, which is meant to denote the relevance of the feature to the target concept. The relief algorithm attempts to find all relevant features. Tree filters [6] use a decision tree algorithm to select a subset of features, typically for a nearest-neighbor algorithm.

The *wrapper approach* has been used to select features of a Bayesian Classifier [21] or for parameter tuning[18]. In [16] a genetic algorithm has been implemented to identify the optimal set of predictive genes that classify samples by cell line or tumor type. Multi-objective approaches have been implemented in several feature selection problems, like handwriting digit recognition [28] and facial expression recognition [35].

7 Conclusions

In this paper we presented a supervised learning module for the optimization of Banksealer, an online banking frauds and anomaly detection framework used by banking analysts as a decision support system. The module was created to solve one of the limitation of the previous version of Banksealer, the inability to collect the feedback given by the analysts and process it to improve the detection performance. The module uses the Non-dominated Sorting Genetic Algorithm II (NSGA-II), a Multi-Objective-Genetic-Algorithm (MOGA), to automatically learn feature weights configurations that optimize the performance of the overall system, instead of relying on manual tuning.

We field-tested the algorithm on real-world data, showing that it is able to self-tune Banksealer over large, unbalanced datasets, and it improves the detection rates over time. The system is extensible and almost transparent to analysts, who just need to express their feedback on the transaction ranking.

Obviously, the system shows some limits that we wish to address in future works. A first experimental limitation is that, while the dataset of transaction is real, in order to create significant tests we needed to inject synthetically generated frauds to evaluate the quality of the detection task. In the future, we will proceed with further tests on real-world fraud samples.

A more relevant limitation is that, as shown in § 5.6, complex fraudulent transactions can still escape the top of the ranking. While we consider the current results already very successful, we believe that the key to improve them further is to design and test other fitness functions and new features (e.g., sum of the amount) that focuses on solving the complex fraud issue. Redesigning fitness functions can also address different motivations for the analysts: for instance, we are experimenting with a fitness function that aims to maximize the total amount of all fraudulent transactions (as opposed to their number). In a MOGA it is rather easy to add and remove fitness functions, and we are going to exploit this modularity in future works.

Acknowledgment

This work has received funding from the European Union's Horizon 2020 Programme, under grant agreement 700326 "RAMSES", as well as from projects co-funded by the Lombardy region and Secure Network S.r.l.

References

1. Kaspersky Security Bulletin 2016. Tech. rep., Kaspersky Lab (2017), https://goo.gl/Jzkab2
2. Almuallim, H., Dietterich, T.G.: Learning with Many Irrelevant Features. In: AAAI. vol. 91, pp. 547–552. Citeseer (1991)
3. Amer, M., Goldstein, M.: Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In: Proc. of the 3rd RapidMiner Community Meeting and Conference (RCOMM 2012). pp. 1–12 (2012)
4. Bolton, R.J., Hand, D.J.: Statistical fraud detection: A review. Statistical Science 17 (2002)
5. Bolton, R.J., Hand, D.J., David J., H.: Unsupervised profiling methods for fraud detection pp. 5–7 (2001)
6. Cardie, C.: Using decision trees to improve case-based learning. In: Proceedings of the tenth international conference on machine learning. pp. 25–32 (1993)
7. Carminati, M., Caron, R., Maggi, F., Epifani, I., Zanero, S.: BankSealer: An Online Banking Fraud Analysis and Decision Support System. In: ICT Systems Security and Privacy Protection. IFIP Advances in Information and Communication Technology, vol. 428, pp. 380–394. Springer Berlin Heidelberg (2014)
8. Carminati, M., Caron, R., Maggi, F., Epifani, I., Zanero, S.: BankSealer: A decision support system for online banking fraud analysis and investigation. Computers & Security 53, 175–186 (2015), http://dx.doi.org/10.1016/j.cose.2015.04.002
9. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. 41(3), 15:1–15:58 (July 2009)
10. Cost, S., Salzberg, S.: A weighted nearest neighbor algorithm for learning with symbolic features. Machine learning 10(1), 57–78 (1993)
11. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Complex Systems 9(3), 1–15 (1994)
12. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. Evolutionary Computation, IEEE Transactions on 6(2), 182–197 (2002)
13. Eiben, A.E., Smith, J.E.: Introduction to evolutionary computing. Springer Science & Business Media (2003)
14. Goldstein, M., Dengel, A.: Histogram-Based Outlier Score (HBOS): A Fast Unsupervised Anomaly Detection Algorithm. KI-2012: Poster and Demo Track pp. 59–63 (2012)
15. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. The Morgan Kaufmann Series in Data Management Systems Series, Elsevier Science & Tech (2006)
16. Jirapech-Umpai, T., Aitken, S.: Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes. BMC bioinformatics 6(1), 148 (2005)
17. Kira, K., Rendell, L.A.: A practical approach to feature selection. In: Proceedings of the ninth international workshop on Machine learning. pp. 249–256 (1992)

18. Kohavi, R., John, G.H.: Automatic parameter selection by minimizing estimated error. In: ICML. pp. 304–312. Citeseer (1995)
19. Kohavi, R., John, G.H.: The wrapper approach. In: Feature Extraction, Construction and Selection, pp. 33–50. Springer (1998)
20. Kononenko, I.: Estimating attributes: analysis and extensions of RELIEF. In: Machine Learning: ECML-94. pp. 171–182. Springer (1994)
21. Langley, P., Sage, S.: Induction of selective Bayesian classifiers. In: Proceedings of the Tenth international conference on Uncertainty in artificial intelligence. pp. 399–406. Morgan Kaufmann Publishers Inc. (1994)
22. Mahalanobis, P.C.: On the generalized distance in statistics. In: Proc. of the national Institute of Science of India. vol. 2, pp. 49–55 (1936)
23. Miller, B.L., Goldberg, D.E.: Genetic algorithms, tournament selection, and the effects of noise. Complex Systems 9(3), 193–212 (1995)
24. Mitchell, M.: An introduction to genetic algorithms. MIT press (1998)
25. Mitchell, T.: Machine Learning. McGraw Hill (1997)
26. Mitra, P., Murthy, C., Pal, S.K.: Unsupervised feature selection using feature similarity. IEEE transactions on pattern analysis and machine intelligence 24(3), 301–312 (2002)
27. Obayashi, S., Takahashi, S., Takeguchi, Y.: Niching and elitist models for MOGAs. In: Parallel Problem Solving from Nature-PPSN V. pp. 260–269. Springer (1998)
28. Oliveira, L.S., Sabourin, R., Bortolozzi, F., Suen, C.Y.: Feature selection using multi-objective genetic algorithms for handwritten digit recognition. In: Pattern Recognition, 2002. Proceedings. 16th International Conference on. vol. 1, pp. 568–571. IEEE (2002)
29. Parks, G.T., Miller, I.: Selective breeding in a multiobjective genetic algorithm. In: Parallel Problem Solving From Nature-PPSN V. pp. 250–259. Springer (1998)
30. Parzen, E.: On estimation of a probability density function and mode. Ann. Math. Statist. 33(3), 1065–1076 (09 1962)
31. Phua, C., Alahakoon, D., Lee, V.: Minority report in fraud detection: classification of skewed data. SIGKDD Explor. Newsl. 6(1), 50–59 (Jun 2004)
32. Phuong, T.M., Lin, Z., Altman, R.B.: Choosing SNPs using feature selection. In: Computational Systems Bioinformatics Conference, 2005. Proceedings. 2005 IEEE. pp. 301–309. IEEE (2005)
33. Punch III, W.F., Goodman, E.D., Pei, M., Chia-Shun, L., Hovland, P.D., Enbody, R.J.: Further Research on Feature Selection and Classification Using Genetic Algorithms. In: ICGA. pp. 557–564 (1993)
34. Rudolph, G.: Evolutionary search under partially ordered sets. Dept. Comput. Sci./LS11, Univ. Dortmund, Dortmund, Germany, Tech. Rep. CI-67/99 (1999)
35. Soyel, H., Tekguc, U., Demirel, H.: Application of NSGA-II to feature selection for facial expression recognition. Computers & Electrical Engineering 37(6), 1232–1240 (2011)
36. Wei, W., Li, J., Cao, L., Ou, Y., Chen, J.: Effective detection of sophisticated online banking fraud on extremely imbalanced data. World Wide Web 16(4), 449–475 (Jul 2013), http://dx.doi.org/10.1007/s11280-012-0178-0
37. Wettschereck, D., Aha, D.W., Mohri, T.: A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. Artificial Intelligence Review 11(1-5), 273–314 (1997)
38. Yang, J., Honavar, V.: Feature subset selection using a genetic algorithm. In: Feature extraction, construction and selection, pp. 117–136. Springer (1998)
39. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary computation 8(2), 173–195 (2000)