Credit Card Fraud Detection: a Realistic Modeling and a Novel Learning Strategy

Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi

***Abstract*—Detecting frauds in credit card transactions is perhaps one of the best testbed for computational intelligence algorithms. In fact, this problem involves a number of relevant challenges, namely: concept drift (customers habits evolve and fraudsters change their strategies over time), class imbalance (genuine transactions far outnumber frauds) and verification latency (only a small set of transactions are timely checked by investigators). However, the vast majority of learning algorithms that have been proposed for fraud detection, relies on assumptions that hardly hold in real-world Fraud Detection Systems (FDS). This lack of realism concerns two main aspects: i) the way and timing with which supervised information is provided and ii) the measures used to assess fraud detection performance.**

This paper has two major contributions. First we propose, with the help of our industrial partner, a formalization of the fraud detection problem which realistically describes the operating conditions of a FDS that everyday analyzes massive stream of credit card transactions. We also illustrate the most appropriate performance measures to be used for fraud detection purposes. Second, we design and assess a learning strategy which effectively address class imbalance, concept drift and verification latency. An additional distinguishing feature of this paper is that the experimental assessment is done on more than 75 millions e-commerce credit card transactions, authorized over a time window of three years.

***Index Terms*—Credit Card Fraud Detection, Unbalanced distribution, Concept Drift, Classification, Learning in nonstationary environments.**

I. Introduction

Credit card fraud detection is a relevant problem that draws the attention of machine-learning and computational intelligence communities, where large number of automatic solutions have been proposed [1], [6], [8], [23], [24], [37], [41], [47], [54], [55], [64]. In fact, this problem appears to be particularly challenging from a learning persepctive, since it is characterized at the same time by *class imbalance* [21], [22], namely genuine transactions far outnumber frauds, and *concept drift* [4], [35], namely transactions might change their statistical properties over time. These, however, are not the only challenges characterizing learning problems in a real-world Fraud-Detection System (FDS).

In a real-world FDS, the massive stream of payment requests is quickly scanned by automatic tools that determine which transactions to authorize. Then, classifiers are typically employed to analyze all the authorized transactions and *alert* the most suspicious ones. Alerts are then inspected by professional investigators that contact the cardholders to determine the true nature (either genuine or fraudulent) of the transaction. By doing this, investigators provide a *feedback* to the system in the form of labeled transactions, which can be used to train (or update) the classifier to preserve (or eventually improve) the fraud detection performance over time. The vast majority of transactions cannot be verified by investigators for obvious time and cost constraints. These transactions remain unlabeled until customers discover and report frauds, or until a sufficient amount of time has elapsed such that non-disputed transactions are considered genuine.

Thus, in practice, most of supervised samples are provided with a substantial delay, a problem known as *verification latency* [44]. The only recent supervised information made available to update the classifier is provided through the *alert-feedback interaction*. Most papers in the literature ignore the verification latency as well as the alert-feedback interaction, and unrealistically assume that the label of each transaction is regularly made available to the FDS, e.g., on a daily basis (see, for instance [6], [8], [12], [23], [24], [28], [47], [54]). However, these aspects have to be considered when designing a real-world FDS, since verification latency is harmful when concept drift occurs, and the alert-feedback interaction is responsible of a sort of *sample selection bias* (SSB) [19] that injects further differences between the distribution of training and test data.

Another important difference between what is typically done in the literature and the real-world operating conditions of FDS concerns the measures used to assess the fraud detection performance. Most often, global ranking measures [23], [24], [61], like the area under the ROC curve (AUC), or cost-based measures [6], [47], [54] are used, but these ignore that only few alerts can be controlled everyday, and that companies are very concerned of the precision of the generated alerts.

The main contributions of this paper are:

- We describe the mechanisms regulating a real-world FDS, and provide a formal model of the articulated classification problem to be addressed in fraud detection.
- We introduce the performance measures that are in practice considered in a real-world FDS.
- Within this sound, realistic, model we propose an effective learning strategy for addressing the above challenges, including the verification latency and the alert-feedback interaction.

Andrea Dal Pozzolo and Gianluca Bontempi are with the Machine Learning Group, Computer Science Department, Faculty of Sciences ULB, Université Libre de Bruxelles, Brussels, Belgium. (email: {adalpozz, gbonte}@ulb.ac.be).

Giacomo Boracchi and Cesare Alippi are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy. (email: {giacomo.boracchi, cesare.alippi}@polimi.it).

Olivier Caelen is with the R&D High Processing & Volume team, Worldline, Belgium. (email: olivier.caelen@worldline.com).

The paper is organized as follows. We first describe in detail the operating conditions of a real-world FDS (Section II), and then (Section III) we model the articulated fraud detection problem and present the most suitable performance measures. In particular, we deem that it is most appropriate to assess the number of detected fraudulent transactions (or cards) over the maximum number of transactions (cards) that investigators can check. The main challenges raising when training a classifier for fraud detection purposes are then discussed in Section IV. Section V introduces the proposed learning strategy, which consists in separately training different classifiers from feedbacks and delayed supervised samples, and then aggregating their predictions. This strategy, inspired by the different nature of feedbacks and delayed supervised samples, is shown to be particularly effective in FDS using sliding window or ensemble of classifiers. We validate our claims in experiments (Section VI) on more than 75 million e-commerce credit card transactions acquired over three years, which are also analyzed to observe the impact of class imbalance and concept drift in real-world transaction streams.

Our work builds upon our previous publication [20], which we significantly extend by describing in detail the real-world operating conditions of a FDS and by analyzing the SSB introduced by the alert-feedback interaction. Furthermore, the experimental section has been largely updated and completed by presenting additional analysis over two large datasets.

II. Real-World FDS

We here describe the main peculiarities and the operating conditions of real-world FDS, inspired by the one routinely used by our industrial partner. Figure 1 illustrates the five layers of control typically employed in a FDS: *i*) the Terminal, *ii*) the Transaction Blocking Rules, *iii*) the Scoring Rules, *iv*) the Data Driven Model (DDM) and *v*) the Investigators.

Layers *i*) - *iv*) implement fully automatic controls, while layer *v*) is the only one requiring human intervention.

A. Layers of Controls in a FDS

1) Terminal: The terminal represents the first control layer in a FDS and performs conventional security checks on all the payment requests [61]. Security checks include controlling the PIN code, the number of attempts, the card status (either active or blocked), the balance availability and the expenditure limit. These operations are performed in real time (response has to be provided in a few milliseconds) in both card and e-commerce transactions. Requests that do not pass any of these controls are denied, while the others become transaction requests that are processed by the second layer of control.

2) Transaction-Blocking Rules: Transaction-blocking rules are *if-then (-else)* statements meant to block transaction requests that clearly resemble frauds. These rules use the few information available when the payment is requested, without analyzing historical records or cardholder profile. An example of blocking rule could be: "*IF internet transactions AND compromised website THEN deny the transaction*"[1]. In practice, several transaction-blocking rules are simultaneously executed, and transactions firing any of these rules are blocked (though cards are not deactivated). Transaction-blocking rules are manually designed by the investigators, and as such are *expert-driven* components of the FDS. To guarantee real-time operations and prevent inconvenience caused when blocking genuine transactions, blocking rules should be: *i*) quick to compute and *ii*) very precise.

All the transactions passing blocking rules are authorized. However, the fraud detection activity continues after having enriched transaction data with aggregated features used to compare the current purchase against the previous ones and the cardholder profile. These aggregated features include, for instance, the average expenditure, the average number of transactions in the same day or the location of the previous purchases. The process of computing aggregated features is referred to as *feature augmentation* and is described in Section II-B. Augmented features and current transaction data are stacked in a *feature vector* that is supposed to be informative for determining whether the authorized transaction is fraudulent or genuine. The following layers of the FDS operate on this feature vector.

3) Scoring Rules: Scoring rules are also expert-driven models that are expressed as *if-then (-else)* statements. However, these operate on feature vectors and assign a score to each authorized transaction: the larger the score, the more likely the transaction to be a fraud.

Scoring rules are manually designed by investigators, which arbitrarily define their associated scores. An example of scoring rule can be "*IF previous transaction in a different continent AND less than one hour from the previous transaction THEN fraud score = 0.95*"[2].

Unfortunately, scoring rules can detect only fraudulent strategies that have already been discovered by investigators, and that exhibit patterns involving few components of the feature vectors. Moreover, scoring rules are rather subjective, since different experts design different rules.

4) Data Driven Model (DDM): This layer is purely data driven and adopts a classifier or other statistical techniques to estimate the probability for each feature vector of being a fraud. This probability is used as the fraud score associated to the authorized transactions. The data-driven model is trained from a set of labeled transactions and can not be interpreted or modified by investigators. An effective data-driven model is expected to detect fraudulent patterns by simultaneously analyzing multiple components of the feature vector, possibly through nonlinear expressions. Therefore, the DDM is expected to find frauds according to rules that go beyond investigator experience, and that do not necessarily correspond to interpretable rules.

This paper focuses on this component of the FDS and proposes a strategy to design, train and update the DDM to improve at best fraud detection performance.

Transactions associated to feature vectors that have either received a large fraud score or an high probability of being

[1]Transaction-blocking rules are confidential and we cannot disclose any rule. Here we provide a realistic example to illustrate which sort of information can be used in these rules.

[2]Scoring rules are also confidential. Our example is meant to illustrate which sort of statements and information a scoring rule might involve.

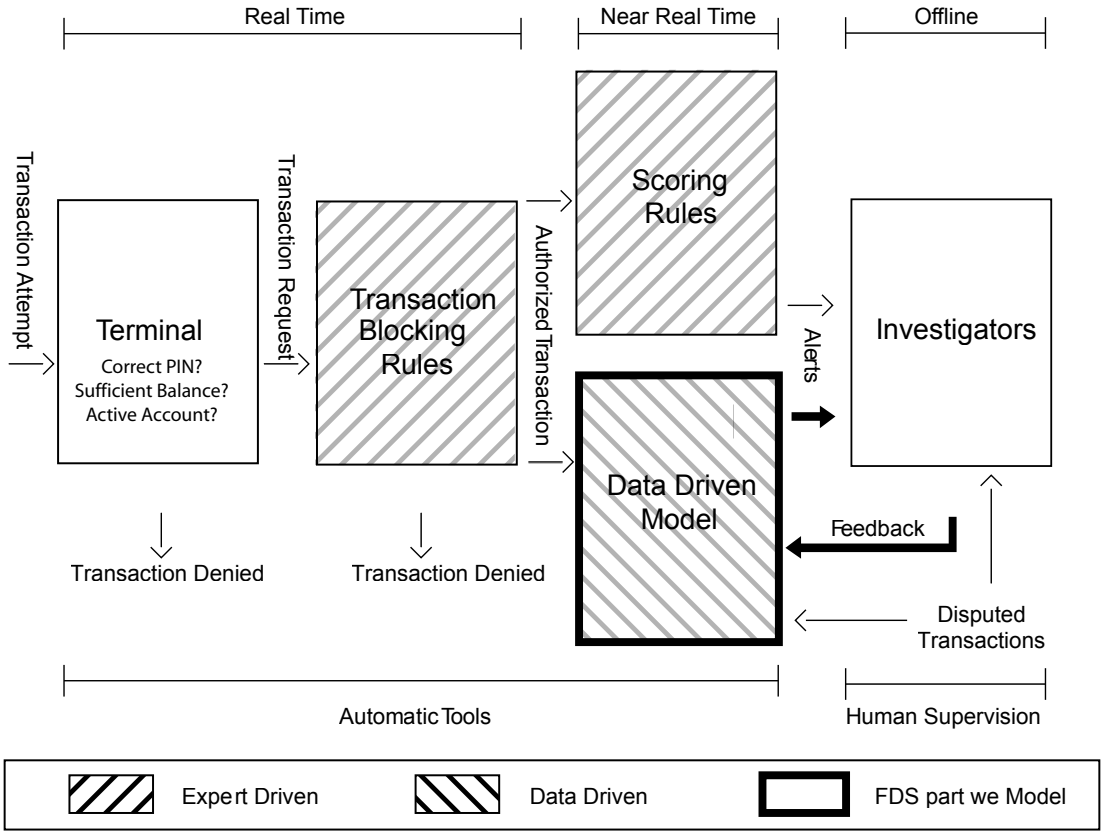


Fig. 1. A scheme illustrating the layers of control in a FDS. Our focus is mainly on the data-driven model and in the *alert-feedback* interaction that regulates the way recent supervised samples are provided.

a fraud, generate alerts. Only a limited number of alerted transactions are reported to the investigators, which represent the final layer of control.

5) Investigators: Investigators are professionals that are experienced in analyzing credit card transactions and are responsible of the expert-driven component of the FDS. In particular, investigators design all the transaction-blocking and scoring rules.

Investigators are also in charge of controlling alerts raised by the scoring rules and the DDM, to determine whether these correspond to frauds or false alarms. In particular, they visualize all the alerted transactions in a case management tool, where all the information about the transaction are reported, including the assigned scores/probabilities, which in practice indicate how *risky* each transaction is. Investigators call cardholders and, after having verified alerted transactions, assign the label "genuine" or "fraudulent", and return this information to the FDS. In the following we refer to these labeled transactions as *feedbacks* and we will use the term *alert-feedback interaction* to describe this mechanism that yield supervised information in a real-world FDS.

Any card that is found victim of a fraud is immediately blocked, to prevent further fraudulent activities. Typically, investigators check all the recent transactions from a compromised card, which means that each detected fraud can potentially generate more than one feedback, which do not necessarily correspond to alerts or frauds.

In a real-world FDS investigators can only check few alerts per day [45] as this process can be long and tedious. Therefore, the primary goal of a DDM is to return precise alerts, as investigators might decide to ignore further alerts when too many false alarms are reported.

B. Features Augmentation

Any transaction request is described by few variables such as the merchant ID, the cardholder ID, the purchase amount, date and time. All transactions requests passing the blocking rules are entered in a database containing all recent authorized transactions, where the feature-augmentation process starts. During feature augmentation, *aggregated* features associated to the authorized transactions are computed to provide additional information about the purchase and better discriminate frauds from genuine transactions. Examples of aggregated features are the average expenditure of the customer every week/month, the average number of transactions per day or in the same shop, the average transaction amount, the location of the last purchases [7], [8], [23], [41], [45], [64]. Van Vlasselaer et al. [61] show that additional informative features can be extracted from the social networks defined by the cardholders and merchants/shops.

Aggregated features are very informative, as they summarize the recent cardholder activities. Thus, they allow to alert transactions that are not suspicious by themselves but might be definitively unusual considering the shopping habits

of each specific cardholder. Features augmentation can be computationally expensive, and aggregated features are often precomputed offline for each cardholder on the basis of historical transactions. Aggregated features are stacked with the transaction data in the feature vector.

C. Supervised Information

Investigators' feedbacks are the most recent supervised information made available to the FDS, but represent only a small fraction of the transactions processed every day [20]. Additional labeled transactions are provided by cardholders that directly dispute unauthorized transactions [20], [61]. The timing of disputed transactions can vary substantially, since cardholders have different habits when checking the transcript of credit card sent by the bank. Moreover, checking disputed transactions entails some necessary administrative procedures that introduce further delays.

All the other transactions remain unlabeled: these can be either genuine transactions or frauds that were missed by the FDS and ignored by the cardholders. However, after a certain time period elapsed without any cardholder dispute, the unreported transactions are by default considered as genuine, and inserted in the training set of the DDM.

Overall, there are two types of supervised information: *i*) feedbacks provided by investigators which are limited in number but refer to recent transactions, and *ii*) delayed supervised transactions, which are the vast majority for which the labels become available after several days (e.g. one month). This latter includes both disputed and non-disputed transactions.

D. System Update

Customers' spending behavior evolves and fraudsters design new attacks, thus their strategies also change over time. It is then necessary to constantly update the FDS to guarantee satisfactory performance. Expert-driven systems are regularly updated by investigators who add ad-hoc (transaction-blocking or scoring) rules to counteract the onset of new fraudulent activities and remove those rules liable of too many false alerts. However, investigators can not modify the DDM, since it is not interpretable and can be only updated (e.g. re-trained) on the basis of recent supervised information, as shown in Figure 1. This operation typically requires a large number of labeled transactions, therefore, though investigators steadily provide feedbacks during the day, the classifier is usually updated/re-trained only once, notably at the end of the day, when a sufficient number of feedbacks is available.

III. Problem formulation

Here we model the articulated classification problem to be addressed in a real-world FDS. In particular, we provide a formal description of the alert-feedback interaction, and present performance measures related to the alert precision. The proposed learning strategy (Section V) and our experiments (Section VI) build upon this model.

Let x_i denote the feature vector associated with the i-th authorized transaction and $y_i \in \{+, -\}$ be the corresponding class, where $+$ denotes a fraud and $-$ a genuine transaction. To cope with the time-variant nature of the transaction stream, a classifier $\mathcal{K}$ is updated (or newly retrained) every day. In particular, we denote by $\mathcal{K}_{t-1}$ the classifier that is trained on supervised transactions available up to day $t-1$. The classifier $\mathcal{K}_{t-1}$ is then used to process the set of transactions T_t that have been authorized at day t. We denote by $\mathcal{P}_{\mathcal{K}_{t-1}}(+|x_i)$ the posterior of $\mathcal{K}_{t-1}$, namely the probability for x_i to be a fraud according to $\mathcal{K}_{t-1}$.

Investigators check only few, high-risk, transactions. Thus, we model *alerts* as the k-most risky transactions, namely

$$\mathrm{A}_t = \{x_i \in \mathrm{T}_t \text{ s.t. } r(x_i) \leq k\}, \tag{1}$$

where $r(x_i) \in \{1, \ldots, |\mathrm{T}_t|\}$ is the rank of x_i according to $\mathcal{P}_{\mathcal{K}_t}(+|x_i)$, and $k > 0$ is the maximum number of alerts that can be checked by investigators[3].

As discussed in Section II-A5, investigators contact the cardholders and provide supervised samples to the FDS in the form of *feedbacks*. In particular, feedbacks include the entirety of recent transactions from the controlled cards, which we model as:

$$\mathrm{F}_t = \{(x_i, y_i) \text{ s.t. } x_i \text{ is from card}(\mathrm{A}_t)\}, \tag{2}$$

where card(A_t) denotes the set of cards having at least a transaction in A_t. If investigators would check only the alerted transactions, feedbacks would be simply defined as $\{(x_i, y_i) \text{ s.t. } x_i \in \mathrm{A}_t\}$.

After a certain verification latency, the labels of all the transactions are provided to the FDS, since, as discussed in Section II-C, non-disputed transactions are considered genuine. For the sake of simplicity, we assume a constant verification latency of δ days, such that at day t the labels of all the transactions authorized at day $t - \delta$ are provided. We refer to these as *delayed supervised samples*:

$$\mathrm{D}_{t-\delta} = \{(x_i, y_i), x_i \in \mathrm{T}_{t-\delta}\}. \tag{3}$$

Note that $\mathrm{F}_{t-\delta} \subset \mathrm{D}_{t-\delta}$ since transactions at day $t-\delta$ obviously include those that have been alerted. Figure 2 illustrates the different types of supervised information available in a FDS.

It is worth mentioning that, despite our formal description includes several aspects and details that have been so far ignored in the fraud detection literature, this is still a simplified model. In fact, alerts in a real-world FDS are typically raised online while transactions are being processed, without having to rank all transactions in T_t. Similarly, the delayed supervised couples do not come all-at-once, as each disputed transactions might take less (or possibly more) than δ days. Notwithstanding, we deem that our formulation takes into account the aspects of a real-world FDS that are the most important ones from a learning perspective, which include alerts, alter-feedback interaction and verification latency.

Fraud detection performance can be conveniently assessed in terms of the *alert precision* $\mathrm{P}_k(t)$, which is defined as:

$$\mathrm{P}_k(t) = \frac{|\mathrm{TP}_k(t)|}{k} \tag{4}$$

[3]Throughout the paper we denote by $|\cdot|$ the cardinality of a set.

where $\text{TP}_k(t) = \{(x_i, y_i)$ such that $x_i \in \text{A}_t$, $y_i = +\}$. Thus, $\text{P}_k(t)$ is the proportion of frauds in the alerts A_t. Though the classifier independently process each feature vector, the alert precision would be more realistically measured in terms of cards rather than authorized transactions. In fact, multiple transactions in A_t from the same card should be counted as a single alert, since investigators check all the recent transactions when contacting cardholders. This implies that k depends on the maximum number of cards that the investigators can control. In this context, it is more informative to measure the detection performance at the card level, such that multiple fraudulent transactions from the same card count as a single correct detection. Thus, we introduce CP_k, the *card precision*, as the proportion of fraudulent cards detected in the k cards controlled by the investigators:

$$\text{CP}_k(t) = \frac{|\text{C}_t^+|}{k}, \tag{5}$$

where C_t^+ denotes the set of fraudulent cards correctly detected at day t, namely, fraudulent cards having reported at least one alert. To correctly account for those days where less than k cards are fraudulent, we define the *normalized* $\text{CP}_k(t)$ as:

$$\text{NCP}_k(t) = \frac{\text{CP}_k(t)}{\Gamma(t)} \quad \text{with} \quad \Gamma(t) = \begin{cases} 1 & \text{if } \gamma_t \geq k \\ \frac{\gamma_t}{k} & \text{if } \gamma_t < k \end{cases} \tag{6}$$

where $\Gamma(t)$ is the maximum value of $\text{CP}_k(t)$ and γ_t is the number of fraudulent cards at day t. From (6) we have that $\text{NCP}_k(t)$ takes values in the range $[0, 1]$, while $\text{CP}_k(t)$ in $[0, 1]$ when $\gamma_t > k$ and in $[0, \frac{\gamma_t}{k}]$ otherwise. For example, if at day t we have correctly detected 40 fraudulent cards ($|\text{C}_t^+| = 40$) out of the $k = 100$ cards checked by investigators, and the overall number of fraudulent cards is 50 ($\gamma_t = 50$), then $\text{CP}_k(t) = 0.4$ while $\text{NCP}_k(t) = \frac{0.4}{0.5} = 0.8$.

Note that, since $\Gamma(t)$ does not depend on the specific classifier $\mathcal{K}_{t-1}$ adopted, if the algorithm "A" is better than algorithm "B" in terms of CP_k, this implies that also "A" will be better than "B" in terms of NCP_k. Let us also remark that the number of fraudulent cards in day t (γ_t) can be computed only after some days (due to verification latency), therefore NCP_k cannot be computed in real time. Thus we recommend using CP_k for assessing running performance, while NCP_k for backtesting e.g. when testing different FDS configurations, as in Section VI-F.

IV. Related works

A. Data Driven Approaches in Credit Card Fraud Detection

Both supervised [8], [12], [15] and unsupervised [11], [14], [60] methods have been proposed for credit card fraud detection purposes. Unsupervised methods consist in outlier/anomaly detection techniques that consider as a fraud any transaction that does not conform with the majority. Remarkably, an unsupervised DDM in a FDS can be directly configured from unlabeled transactions. A well known method is Peer Group Analysis [63], which clusters customers according to their profile and identifies frauds as transactions departing from the customer usual behaviour (see also the recent survey by Phua et al. [52]). Typical cardholder's behavior have also been modeled by means of self-organizing maps [51], [53], [69].

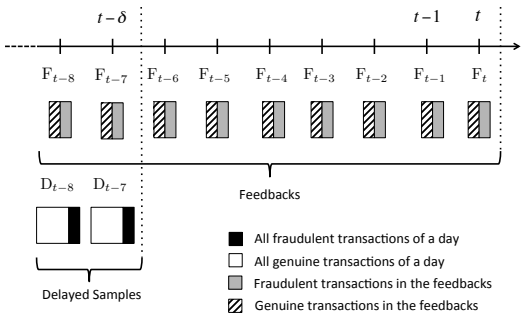


Fig. 2. The supervised samples available at the end of day t include: *i*) feedbacks ($F.$) and *ii*) delayed couples ($D.$) occurred before the $t - \delta^{th}$ day. Here we assume $\delta = 7$. Patterns indicate different labels, and the size of these regions indicates class proportions.

Supervised methods are by far the most popular in fraud detection, and exploit labeled transactions for training a classifier. Frauds are detected by classifying feature vectors of the authorized transactions or possibly by analyzing the posterior of the classifier [10]. Several classification algorithms have been tested on credit card transactions to detect frauds, and in particular Neural Networks [1], [12], [28], [37], Logistic Regression [41], Association Rules [55], Support Vector Machines [64], Modified Fisher Discriminant Analysis [47], and Decision Trees [6], [24], [54]. However, several studies have reported Random Forest to achieve the best performance [8], [20], [23], [61], [64]. This is one of the reasons why we adopt Random Forests in our experiments.

B. Performance Measure for Fraud Detection

The typical performance measure for fraud detection problems is the area under the ROC curve (AUC) [23], [24], [61]. AUC can be estimated by means of the Mann-Whitney statistic [48] and its value can be interpreted as the probability that a classifier ranks frauds higher than genuine transactions [38]. Another ranking measure frequently used in fraud detection is Average Precision (AP) [23], which corresponds to the area under the precision-recall curve. While these measures are meant for detection problems in general, cost-based measures are specifically designed for fraud detection purposes. Cost-based measures [6], [47], [54] quantify the monetary loss of each fraud by means of a cost matrix that associates a cost to each entry of the confusion matrix. Elkan [29] warns against the risk of using different baselines when using a cost-matrix to measure the overall cost, because the minimum/maximum loss can change over time. To avoid this problem, *normalized* cost [64] or savings [6] are used to asses the performance w.r.t. the maximum loss.

We argue that performance measures should also account for the investigators availability, as they have to check all the alerts raised by the FDS. Given the limited time investigators have, only a few alerts can be verified every day, thus an effective FDS should provide investigators a small number of reliable alerts. For this reason, we have introduced the alert-precision measures described in Section III.

C. Major Challenges to be Addressed in a real-world FDS

As anticipated in Section I, the major challenges to be addressed when designing a FDS include: *i*) handling the class *imbalance*, since legitimate transactions far outnumber the fraudulent ones, *ii*) handling the *concept drift* since the statistical properties of both frauds and genuine transactions evolve with time and *iii*) operating with a small number of recent supervised transactions, provided in the form of investigators' feedback.

1) Class Imbalance: Class distribution is extremely unbalanced in credit card transactions, since frauds are typically less than 1% of the overall transactions, as shown in [24], [45] and in our analysis (see Table I). Learning under class imbalance has received a lot of attention in the computational intelligence and machine-learning community, because traditional learning methods yield classifiers that are poorly performing on the minority class, which is definitively the class of interest in detection problems.

Several techniques have been proposed to deal with class imbalance, and for a comprehensive overview we refer the reader to the survey [39]. The two main approaches for dealing with class imbalance are: *i*) *sampling* methods and *ii*) *cost-based* methods. Sampling methods are used to balance the class distribution in the training set before running a traditional learning algorithm, while cost-based methods modify the learning algorithm to assign a larger misclassification cost to the minority class [29].

Sampling methods are divided in undersampling ones, which balance the class proportions in the training set by removing samples from the majority class, and oversampling, which achieve the same goal by replicating training samples of the minority class [21]. Advanced oversampling methods like SMOTE [17] generate synthetic training instances from the minority class by interpolation, instead of sample replication.

Cost-based methods do not need to balance the proportion of training data, as they take into account different losses for classification errors on the minority and majority samples. In credit card fraud detection, the cost of a missed fraud is often assumed to be proportional to the transaction amount [6], [47], [54], and this assigns a larger misclassification cost to frauds, thus steering the classifier to prefer false alerts rather than taking the risk of missing a fraud. As a consequence, these algorithms might generate many false positives while investigators require precise alerts.

2) Concept Drift: There are two main factors introducing changes/evolutions in the stream of credit card transactions, which in the literature are typically referred to as concept drift [27], [34]. At first, genuine transactions evolve because cardholders typically change their spending behaviors over time (e.g., during holidays they purchase more and differently from the rest of the year). Second, also frauds change over time, since new fraudulent activities are perpetrated. In our experiments (see Section VI-D) we observe the evolving nature of credit card transactions in two large datasets of real-world e-commerce transactions. Learning under concept drift is one of the major challenges that data-driven methods have to face, since classifiers operating in these conditions have in practice to autonomously identify the most relevant, up-to-date, supervised information while ignoring the obsolete one. Concept drift adaptation approaches can be divided in two families: *i*) active adaptation and *ii*) passive adaptation.

Active approaches [4], [9], [33], [50] use a change-detection test [3] or other statistical triggers to monitor the incoming data by analyzing the classification error and/or the data distribution [2]. As soon as a change in the incoming data is detected, adaptation is activated and the classifier is updated/retrained on those recent supervised samples that are considered to be coherent with the current state of the process. As such, active approaches are mostly suited when the data distribution changes abruptly, and the process generating the data shifts through a sequence of stationary states.

In passive approaches, the classifier is continuously updated when new supervised samples become available, without involving any explicit triggering mechanism. Ensemble methods [23], [30], [43], [59], [70], and classifiers trained over a sliding window of the recent supervised samples (like STAGGER [56] and FLORA [65]) are probably the most extensively investigated passive solutions. Passive approaches are the more suitable ones in gradually drifting environments, and when the supervised information is provided in batches.

When data streams are characterized by both concept drift and unbalanced distributions, adaptation is often achieved by combining ensemble methods and resampling techniques [26], [36], [62]. An alternative approach consists in propagating the training samples of the minority class over time [36], possibly undersampling the majority class. Chen and He proposed REA [18] which only propagates examples from the minority class that belongs to the current concept.

3) Alert-Feedback Interaction and Sample Selection Bias: The majority of classifiers used for credit card fraud detection in the literature (e.g. [11], [12], [15]) are tested in experiments where transaction labels are supposed to be available the very next day after the transaction authorization. In a real-world FDS (Section II-C) the only recent supervised information are the feedbacks F_t, provided by investigators, while the vast majority of transactions authorized everyday does not receive a label in a short time. The few feedbacks provided ($|\mathrm{F}_t| << |\mathrm{T}_t|$) are not representative of the transactions processed everyday for two main reasons: *i*) feedbacks contain transactions that are characterized by a high probability of being frauds, and *ii*) the proportion of frauds in the feedbacks is different from the proportion of frauds occurring everyday. Thus, feedbacks represent a sort of biased training set: this problem evokes what is known in the literature as Sample Selection Bias (SSB) [19].

A biased training set may hinder the performance of learning algorithms, since training data do not match the distribution of the test ones. The reader can refer to [49] for a survey on SSB. Here we simply mention that there are three different types of SSB: class-prior bias, feature bias (also called covariate shift) and complete bias. A standard remedy to SSB is *importance weighting* [32], [67], [68], namely semi-supervised re-weighting techniques that assign larger weights to those training samples that more closely resamble the data distribution in the test set. The main idea of importance

weighting is to reduce the influence of the most biased samples in the learning process. Ensembles of classifiers have been also proposed to correct SSB [31].

The interaction between the FDS (raising alerts) and the investigators (providing true labels) recalls the active learning scenario [57], where the classifier is allowed to select few –very informative– samples and query their labels to an oracle which in the FDS would be the investigators. However, this is not feasible in a real-world FDS, since investigators have to focus on the most suspicious transactions to detect the largest number of frauds. Requests to check (possibly genuine) transactions for obtaining informative samples would be ignored. In fact, given the limited number of transactions investigators can check, these queries imply that some high-risk transaction would not be controlled, with the consequent loss in detection performance.

V. Learning Strategy

It is important to stress that supervised samples belonging to feedbacks (F_t) and to delayed samples ($\mathrm{D}_{t-\delta}$) feature very different characteristics. The first difference is quite evident: F_t provides recent, up-to-date, information while $\mathrm{D}_{t-\delta}$ might be already obsolete for training a classifier that is meant to analyze transactions that will be authorized the next day. The second difference concerns the percentage of frauds in F_t and $\mathrm{D}_{t-\delta}$: while the class proportion in $\mathrm{D}_{t-\delta}$ is heavily skewed towards the genuine class (see the proportions of frauds in Table I), the number of frauds in F_t actually depends on the detection performance of $\mathcal{K}_{t-1}$, and high precision values might even result in F_t skewed towards frauds. The third, and probably the most subtle, difference is that supervised couples in F_t are not independently drawn, but are instead transactions from cards selected by $\mathcal{K}_{t-1}$ as those that are most likely to have been frauded. As such, F_t is affected by SSB and any classifier trained on F_t would in principle learn how to label transactions that are most likely to be fraudulent. Thus, this might not be in principle precise on the vast majority of genuine transactions.

Our intuition is that feedbacks and delayed samples are representative of two different classification problems, thus they have to be separately handled. Therefore, our learning strategy consists in training a classifier exclusively on feedbacks (i.e., $\mathcal{F}_t$) and a classifier exclusively on delayed supervised samples (i.e., $\mathcal{D}_t$), and by aggregating their posterior probabilities when defining $\mathcal{P}_{\mathcal{K}_t}(+|x_i)$ to determine which transactions to alert.

In the following we detail the proposed learning strategy, where adaptation is performed according to a passive approach and the classifier is updated everyday on a batch containing the latest supervised couples available, either feedbacks or delayed samples. As in Section III, we consider a constant verification latency of δ days. In particular, to process the transactions authorized at day $t+1$, we rely on Q days of feedbacks $\{\mathrm{F}_t, \ldots, \mathrm{F}_{t-(Q-1)}\}$, and M days of delayed supervised samples $\{\mathrm{D}_{t-\delta}, \ldots, \mathrm{D}_{t-(\delta+M-1)}\}$, and these latter obviously include the feedbacks received in the same days (i.e., $\mathrm{F}_i \subset \mathrm{D}_i,\ i \leq t-\delta$). Our learning strategy, which is detailed in Algorithm 1, consists in separately training a classifier $\mathcal{F}_t$ on feedbacks

$$\mathcal{F}_t = \text{TRAIN}\left(\{\mathrm{F}_t, \ldots, \mathrm{F}_{t-(Q-1)}\}\right) \tag{7}$$

and a classifier on delayed supervised samples

$$\mathcal{D}_t = \text{TRAIN}\left(\{\mathrm{D}_{t-\delta}, \ldots, \mathrm{D}_{t-(\delta+M-1)}\}\right) \tag{8}$$

and to detect frauds by the *aggregation* $\mathcal{A}_t$, whose posterior probability is defined as:

$$\mathcal{P}_{\mathcal{A}_t}(+|x) = \alpha \mathcal{P}_{\mathcal{F}_t}(+|x) + (1-\alpha)\mathcal{P}_{\mathcal{D}_t}(+|x) \tag{9}$$

where $0 \leq \alpha \leq 1$ is the weight parameter that balances the contribution of $\mathcal{F}_t$ and $\mathcal{D}_t$. Thus, the posterior probability of the classifier $\mathcal{K}_t$, which alerts the transactions authorized at day $t+1$, is given by (9). Note that to train $\mathcal{F}_t$ we can also use feedbacks received before δ days ($Q \geq \delta$) and in particular we use $Q = \delta + M$.

The rationale behind the proposed learning strategy is two-fold. At first, by training a classifier (7) exclusively on feedbacks, we guarantee larger relevance to these supervised samples, which would be otherwise outnumbered by the delayed supervised samples. Second, we alert only those transactions that both $\mathcal{F}_t$ and $\mathcal{D}_t$ consider most probably frauds: this follows from the fact that, in practice, because of the large number of transactions processed everyday, alerts corresponds to values of $\mathcal{P}_{\mathcal{A}_t}$ that are very close to 1. Let us recall that $\mathcal{F}_t$, thus also $\mathcal{A}_t$, is affected by SSB due to alert-feedback interaction. The only training samples that are not affected by SSB are the delayed supervised samples which, however, might be obsolete because of concept drift.

Algorithm 1 Proposed Learning Strategy

Require: M and Q, i.e., the number of days of delayed samples and feedbacks to use, respectively; $\mathcal{F}_t$ and $\mathcal{D}_t$ classifiers previously trained.
$\mathrm{T}_{t+1} \leftarrow$ transactions at day $t+1$.
for each transaction $x \in \mathrm{T}_{t+1}$ **do**
  compute $\mathcal{P}_{\mathcal{F}_t}(+, x)$
  compute $\mathcal{P}_{\mathcal{D}_t}(+, x)$
  compute $\mathcal{P}_{\mathcal{A}_t}(+, x)$ as in (9)
rank T_{t+1} according to $\mathcal{P}_{\mathcal{A}_t}(+, \cdot)$,
generate alerts A_t.
if update the classifier **then**
  $\mathrm{F}_{t+1} \leftarrow$ feedbacks from cards alerted in A_t.
  $\mathcal{F}_{t+1} \leftarrow$ TRAIN($\{\mathrm{F}_{t+1}, \ldots, \mathrm{F}_{t-Q}\}$)
  $\mathrm{D}_{t+1-\delta} \leftarrow$ transactions authorized at $t+1-\delta$
  $\mathcal{D}_{t+1} \leftarrow$ TRAIN($\{\mathrm{D}_{t+1-\delta}, \ldots, \mathrm{D}_{t-(\delta+M)}\}$)
return $\mathcal{F}_t, \mathcal{D}_t$ and $\mathcal{A}_t$ defined as in (9).

A. Implementation of the Proposed Learning Strategy

In our experiments we implement the proposed learning strategy in two different scenarios, which correspond to two mainstream approaches for learning $\mathcal{D}_t$. In the former, $\mathcal{D}_t$ is a *sliding window* classifier as in [60], [61], which we denote by $\mathcal{W}_t^D$, while in the latter $\mathcal{D}_t$ is an *ensemble* of

classifiers similar to [23], [36], which we denote by $\mathcal{E}_t^D$. Both the classifiers $\mathcal{W}_t^D$ and $\mathcal{E}_t^D$ are trained on delayed samples $\{\mathrm{D}_{t-\delta},\ldots,\mathrm{D}_{t-(\delta+M-1)}\}$. However, while $\mathcal{W}_t^D$ employs a unique model to this purpose, $\mathcal{E}_t^D$ is an ensemble of M classifiers $\{\mathcal{M}_1,\mathcal{M}_2,\ldots,\mathcal{M}_M\}$ where each individual classifier $\mathcal{M}_i$ is trained on delayed samples of a different day, i.e., $\mathrm{D}_{t-\delta-i}, i=0,\ldots,M-1$. The posterior $\mathcal{P}_{\mathcal{E}_t^D}(+|x)$ is obtained by averaging the posterior probabilities of the individual classifiers, i.e., $\mathcal{P}_{\mathcal{E}_t^D}(+|x)=\frac{\sum_i^M \mathcal{P}_{\mathcal{M}_i}(+|x)}{M}$.

In the sliding window case, the proposed learning strategy consists in analyzing the posterior of the classifier $\mathcal{A}_t^W$, which aggregates $\mathcal{F}_t$ and $\mathcal{W}_t^D$, i.e., $\mathcal{P}_{\mathcal{A}_t^W}(+|x) = \alpha\mathcal{P}_{\mathcal{F}_t}(+|x)+(1-\alpha)\mathcal{P}_{\mathcal{W}_t^D}(+|x)$ as in (9). The benchmark to compare against $\mathcal{A}_t^W$ is the classifier $\mathcal{W}_t$, which is trained on all the supervised transactions referring to the same time interval (thus mixing delayed samples and feedbacks): $\{\mathrm{F}_t,\ldots,\mathrm{F}_{t-(\delta-1)},\mathrm{D}_{t-\delta},\ldots,\mathrm{D}_{t-(\delta+M-1)}\}$.

Similarly, in the ensemble case, the proposed learning strategy consists in analyzing the posterior of the classifier $\mathcal{A}_t^E$, which is obtained by aggregating the posteriors of $\mathcal{F}_t$ and $\mathcal{E}_t^D$, i.e., $\mathcal{P}_{\mathcal{A}_t^E}(+|x)=\alpha\mathcal{P}_{\mathcal{F}_t}(+|x)+(1-\alpha)\mathcal{P}_{\mathcal{E}_t^D}(+|x)$, as in (9). The benchmark to compare against $\mathcal{A}_t^E$ is the classifier $\mathcal{E}_t$, whose individuals are $\{\mathcal{M}_1,\mathcal{M}_2,\ldots,\mathcal{M}_M,\mathcal{F}_t\}$, and whose posterior $\mathcal{P}_{\mathcal{E}_t}(+|x)$ is estimated by averaging the posterior probabilities of all its individuals, i.e., $\mathcal{P}_{\mathcal{E}_t}(+|x)=\frac{\sum_i^M \mathcal{P}_{\mathcal{M}_i}(+|x)+\mathcal{P}_{\mathcal{F}_t}(+|x)}{M+1}$.

In both aggregations $\mathcal{A}_t^W$ and $\mathcal{A}_t^E$ we set $\alpha=0.5$ to give equal contribution to the feedback and delayed classifier, as better discussed in Section VI-F.

For all the base classifiers involved (i.e., $\mathcal{F}_t,\mathcal{W}_t^D,\mathcal{W}_t,\mathcal{M}_i,i=1,\ldots,M$) we adopt Random Forest (RF) [13] having 100 tree each. Each tree is trained on a balanced bootstrap sample, obtained by randomly undersampling the majority class while preserving all the minority class samples in the corresponding training set. In this way, each tree is trained on randomly selected genuine transactions and on the same fraud examples. This undersampling strategy allows one to learn trees with balanced distribution and to exploit many subsets of the majority class. At the same time, the training times of these classifiers are reasonably low. A drawback of undersampling is that we potentially remove relevant training samples from the dataset, though this problem is mitigated by the fact that we learn 100 different trees for each base classifier.

VI. Experiments

Our experiments are organized as follows: in Section VI-A we describe our datasets and in Section VI-B we illustrate few details about the performance measures. Section VI-C presents our first experiment which uses the classifiers in Section V-A to assess the effectiveness of the proposed learning strategy. In the second experiment (Section VI-D) we analyze more than 54 Millions of credit card transactions acquired over 10 months, and show that this stream is seriously affected by concept drift. Then, to investigate the adaptation ability of the proposed learning strategy, we synthetically introduce an abrupt concept drift in specific locations of the transaction stream, and assess the classification performance.

In the third experiment (Section VI-E) we investigate the sample-selection bias introduced by the alert-feedback interaction, and we show that importance weighting [19] – a conventional technique to correct SSB – is not effective on training sets of feedbacks. Finally, in Section VI-F we discuss the most important parameters influencing the proposed learning strategy.

A. Our Datasets

We use two large dataset of e-commerce transactions from European credit card holders, provided by our industrial partner. Even though these transactions are not initiated from a physical terminal, they undergo the same process described in Figure 1, including PIN control in some cases. The first dataset, referred to as 2013, contains transactions from September 2013 to January 2014 and is the same used in [20]; the second one, referred to as 2014-2015, contains transactions from August 2014 to May 2015.

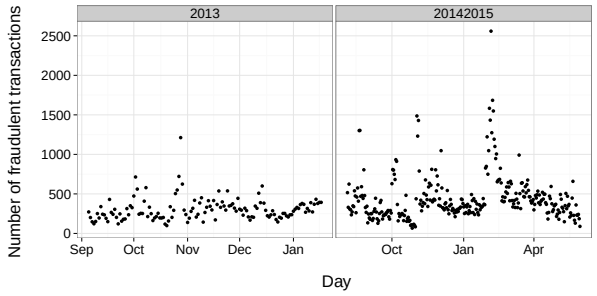
Table I provides additional information about these datasets, showing the extreme class-imbalance ratio, since frauds account for about 0.2% of all transactions. The number of frauds per day varies significantly over time, as shown in Figure 3(a). Figure 3(a) also shows that the number of fraudulent transactions is larger than that of fraudulent cards, indicating that multiple frauds are sometimes perpetrated on the same cards. Part of 2013 dataset have been suitably anonymized and made publicly available for download in [22].

TABLE I
DATASETS

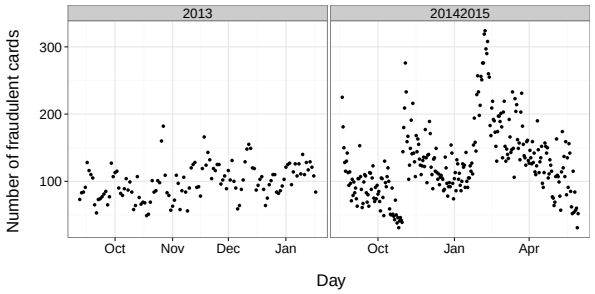
Id	Start day	End day	# Days	# Instances	# Features	% Fraud Trx
2013	2013-09-05	2014-01-18	136	21'830'330	51	0.19%
2014-2015	2014-08-05	2015-05-31	296	54'764'384	51	0.24%

To reliably assess the fraud detection performance in terms of P_k, we have removed the *CARD_ID* component from all the feature vectors. This is very important when testing a classifier on a dataset of historical transactions, since a classifier that receives in input the variable *CARD_ID* might learn this as a discriminative feature to detect multiple frauds from the same card in different days (thus providing too optimistic performance). However, in a real-world FDS, it is not possible to have multiple frauds from the same card after having detected the first one since, as discussed in Section II, that card would have been immediately blocked. The other option would be to remove all the transactions of the same card after having detected the first fraud. However, this would have reduced the number of frauds available, further worsening the class imbalance in our dataset. Therefore, we decided to consider the *CARD_ID* feature to compute the aggregated features, and then not to include it in the feature vectors.

It is important to stress that cards having multiple transactions rated as high-risky ones (from either the DDM or the Scoring Rules) clearly deserve particular attention, and that these aspects should be considered in a real-world FDS. While this information cannot be straightforwardly exploited

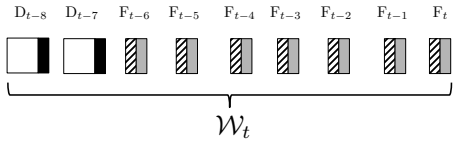


(a) Number of fraudulent transactions

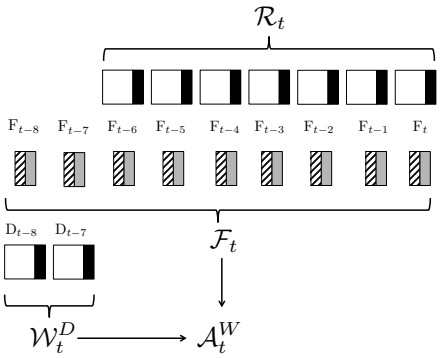


(b) Number of fraudulent cards

Fig. 3. Number of fraudulent transactions and cards per day in the datasets illustrated in Table I. It emerges there are more fraudulent transactions than cards, meaning that some fraudulent cards have received more than a fraud.



(a) Pooling together all labeled transactions



(b) Separating feedbacks and delayed samples

Fig. 4. Supervised information used by different classifiers when $\delta = 7$, $M = 2$ and $Q = 7 + 2 = 9$.

TABLE II
CLASSIFIERS CONSIDERED IN OUR EXPERIMENTS.

Symbol	supervised samples	adaptation	# days training
$\mathcal{F}$	feedbacks	sliding	Q
$\mathcal{W}^D$	delayed	sliding	M
$\mathcal{W}$	feedbacks + delayed	sliding	$\delta + M$
$\mathcal{A}^W$	feedbacks + delayed	sliding	$Q + M$
$\mathcal{R}$	all the recent	sliding	δ
$\mathcal{E}^D$	delayed	ensemble	M
$\mathcal{E}$	feedbacks + delayed	ensemble	$\delta + M$
$\mathcal{A}^E$	feedbacks + delayed	ensemble	$Q + M$

by a DDM that independently analyzes each feature vector, it can be easily exploited in a scoring rule or included in a suitable component of the feature vector during the feature augmentation process.

B. Experimental Settings

In agreement with our industrial partner, we assumed that investigators can check transactions from 100 cards alerted by the DDM every day. Thus, $\mathcal{F}_t$ is trained on transactions from 100 different cardholders whose transactions have been previously alerted. Let us recall that feedbacks depend on the actual classifier requesting the labels. As such, the training set of $\mathcal{F}_t$ might be different when used in $\mathcal{A}_t$ and when used standalone: in fact, in the former case alerts depend also on the posterior of $\mathcal{D}_t$, while in the latter, alerts are uniquely determined by $\mathcal{F}_t$.

The overall FDS performance is assessed by averaging P_k or CP_k over different days, and when considering average performance, we omit the index t from the classifier notation. We then use a paired t-test to assess whether the performance gaps between different classifiers are statistically significant. To this purpose, we pursue the approach recommended by Demsar [25], which relies on paired t-test on the ranks resulting from the Friedman test. For each day in our dataset, we rank the classifiers from the least to the best performing one and we compare classifier performance by means of a paired t-test on the ranks, and we sum the ranks over all days. More formally, let us consider S classifiers to compare, such that $r_{\mathcal{K},j} \in \{1, \ldots, S\}$ is the rank of the classifier $\mathcal{K}$ on day j. In those days j when $\mathcal{K}$ is the best classifier, then $r_{\mathcal{K},j} = S$, while when it is the worst $r_{\mathcal{K},j} = 1$. The sum of ranks for classifier $\mathcal{K}$ is $\sum_{j=1}^{J} r_{\mathcal{K},j}$, the higher this sum, the more often classifier $\mathcal{K}$ results superior to the others. The paired t-test compares the ranks of two classifiers $\mathcal{K}$ and $\mathcal{H}$ over all days (i.e., $r_{\mathcal{K},j} - r_{\mathcal{H},j}, j \in \{1, \ldots, J\}$) where J is the number of considered days[4].

Each experiment is repeated 10 times to reduce the variability introduced by the boostrap procedure used to train the Random Forests. In most of our experiments we consider a verification latency of one week ($\delta = 7$), then in Section VI-F we repeat the experiments considering a verification latency of 15 days ($\delta = 15$).

C. Separating Feedbacks From Delayed Supervised Samples

To assess the effectiveness of the proposed learning strategy, we compare the performance of the proposed classifiers $\mathcal{A}^W$ (resp. $\mathcal{A}^E$) against the corresponding benchmarks introduced in Section V-A and the classifiers used to define their posteriors, i.e. $\mathcal{F}$ and $\mathcal{W}^D$ (resp. $\mathcal{E}^D$). Figure 4 illustrates the training

[4]Since the training sets used in consecutive days are largely overlapping, the ranks are not independent and this might affect the significance of the tests. In fact, when a classifier is outperforming others in one day, it is also likely to do the same during the next few days. However, this is a standard post-hoc analysis for non-parametric tests, such as the Friedman test [25].

set involved when using $\mathcal{A}_t^W$ and the related classifiers, while Table II summarizes the most important parameters and the training samples used by the considered classifiers.

In this experiment we have also included the classifier $\mathcal{R}_t$ which is trained on all transactions authorized between day t and $t-\delta$. This classifier is sort of *ideal* counterpart of sliding window classifiers (thus it is compared against $\mathcal{A}^W$), as this underlines the unrealistic assumption that investigators can everyday assign the correct label to each authorized transaction. In particular, the training set of $\mathcal{R}_t$ is not influenced by the alert-feedback interaction.

Table III shows the average P_k, CP_k and AUC over all the batches for the two datasets separately. The columns *comparison* report the results of the paired t-test on the ranks described above. Classifiers having the same letter cannot be considered significantly different. In both datasets, $\mathcal{A}^W$ outperforms $\mathcal{W}$ in terms of P_k and CP_k, and this indicates that separating feedbacks and delayed samples is indeed a good learning strategy. The same result holds for the considered ensembles, i.e., $\mathcal{A}^E$ and $\mathcal{E}$. Since both $\mathcal{A}^E$ and $\mathcal{E}$ average the posteriors of their individuals, their difference consists only in the aggregation weights: in $\mathcal{A}^E$, 50% of the total weight is assigned to $\mathcal{P}_{\mathcal{F}}(+|x)$ and the remaining 50% is equally distributed to the other individuals, while in $\mathcal{E}$ all the individuals contribute equally. The same relation does not hold between $\mathcal{A}^W$ and $\mathcal{W}$, which are updated in a sliding-window manner. However, also in this case we can conclude that feedbacks are very informative and need to be carefully considered to increase the alert precision. This is also confirmed by the fact that $\mathcal{F}$ outperforms both $\mathcal{W}^D$ and $\mathcal{W}$. As a general comment, we notice that CP_k is typically lower than P_k, since often, multiple frauds are perpetrated on the same card.

Table III reports also the results in terms of AUC, which is a global ranking measure as it evaluates the classifier's posteriors over all the instances and not only in the top k ranking ones (differently from CP_k and P_k). As expected, the ideal classifier $\mathcal{R}$ is significantly better than $\mathcal{A}^W$ in terms of AUC, indicating that it achieves superior ranking performance over the whole set of transactions. The performance of $\mathcal{F}$ is by far the worse in terms of AUC, as this classifier is not able to provide a suitable ranking to all the transactions.

We interpret these results as follows: when the goal is to get accurate ranking of the most suspicious cards (i.e., maximize CP_k) we should give larger weights to those transactions that are as risky as the one we want to predict, hence use $\mathcal{A}^W$. On the contrary, a classifier trained on all daily transactions (which are mostly genuine) is better at ranking all the transactions, as it emerges from the AUC. In Table III we also see that $\mathcal{R}$ outperforms $\mathcal{W}^D$ in terms of both P_k, CP_k and AUC. This fact indicates that the stream of credit card transactions is nonstationary, since when comparing two identical classifiers, the one trained on more recent transactions achieves superior performance.

D. Concept Drift

In this section we first analyze the 2014-2015 dataset which contains more than 54 millions transaction authorized over ten months and show that this stream is affected by concept drift. To this purpose, we use a static classifier $\mathcal{S}_t$, which is initially trained on M days and never updated[5] and compare it against $\mathcal{W}_t^D$ (which is regularly updated) and $\mathcal{A}_t^W$ (which leverages feedbacks as well). In a stationary classification problem, the two classifiers $\mathcal{S}$ and $\mathcal{W}^D$ would perform similarly. The fact that over time $\mathcal{S}_t$ outperforms $\mathcal{W}_t^D$ (see Figure 5.a) confirms that this dataset is affected by concept drift. While it might not sound surprising that the stream of credit card transactions is nonstationary, this is, to the best of our knowledge, the first analysis on the impact of concept drift on such a large transaction dataset.

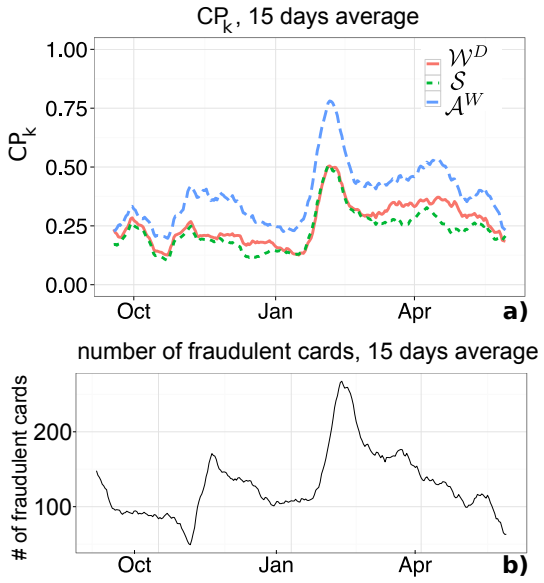


Fig. 5. a) The values of CP_k for $\mathcal{S}$, $\mathcal{W}^D$ and $\mathcal{A}^W$ on dataset 2014-2015; b) the number of frauded credit cards on the same period. For the visualization sake these values have averaged over a sliding window of 15 days.

Figure 5.a also shows that the proposed $\mathcal{A}^W$ always achieves superior performance in terms of CP_k, demonstrating a better adaptation to concept drift. It is worth noting that the performance of all the classifiers in Figure 5.a fluctuate quite heavily and report a peak during February 2015. This is indeed the month having the largest number of fraudulent cards in our dataset (which is reported in Figure 5.b). In contrast, during October 2014 (the period exhibiting the lowest number of fraudulent cards in our dataset) all the classifiers achieve low values of CP_k. Thus, Figure 5 confirms that the alert precision heavily depends on the number of fraudulent cards in a day.

To investigate further the adaptation performance of $\mathcal{A}^W$ in a nonstationary environment, we preform an additional experiment where we artificially introduce an abrupt concept drift at known locations. As in [20], we prepared 10 short datasets by juxtaposing transactions authorized in two non consecutive months. Thus, each of these datasets contains an abrupt CD in the middle, which should be more clearly perceivable when the time distance between the juxtaposed months increases. To assess the adaptation ability of the proposed learning strategy, we compare the performance of

[5]Thus, $\mathcal{S}_t$ initially coincides with $\mathcal{W}_t^D$.

TABLE III
FRAUD DETECTION PERFORMANCE WHEN USING 15 DAYS OF TRANSACTIONS ($\delta = 7$, $M = 8$, $Q = 15$)

Classifier	Dataset	Average P_k			Average CP_k			Average AUC		
		mean (std)	sum of ranks	comparison	mean (std)	sum of ranks	comparison	mean (std)	sum of ranks	comparison
$\mathcal{A}^W$	2014-2015	0.77 (0.21)	1796.50	a	0.37 (0.18)	1824.00	a	0.94 (0.02)	1396.00	b
$\mathcal{F}$	2014-2015	0.73 (0.23)	1632.00	b	0.32 (0.17)	1505.00	b	0.87 (0.05)	409.00	e
$\mathcal{R}$	2014-2015	0.63 (0.24)	1156.00	c	0.30 (0.18)	1354.50	c	0.96 (0.02)	1822.00	a
$\mathcal{W}$	2014-2015	0.61 (0.25)	1055.50	d	0.25 (0.14)	955.00	d	0.91 (0.04)	865.00	d
$\mathcal{W}^D$	2014-2015	0.57 (0.26)	889.00	e	0.25 (0.14)	885.00	e	0.94 (0.03)	1315.00	c
$\mathcal{A}^W$	2013	0.75 (0.20)	732.00	a	0.35 (0.12)	754.50	a	0.94 (0.03)	631.00	b
$\mathcal{F}$	2013	0.73 (0.21)	693.00	b	0.32 (0.13)	670.50	b	0.89 (0.05)	229.00	e
$\mathcal{R}$	2013	0.58 (0.22)	493.50	c	0.25 (0.11)	514.00	c	0.96 (0.01)	736.00	a
$\mathcal{W}$	2013	0.54 (0.25)	434.00	d	0.22 (0.11)	387.00	d	0.91 (0.05)	355.00	d
$\mathcal{W}^D$	2013	0.50 (0.23)	345.00	e	0.21 (0.09)	330.00	e	0.93 (0.03)	539.00	c
$\mathcal{A}^E$	2014-2015	0.77 (0.21)	981.50	a	0.39 (0.17)	940.00	a	0.94 (0.03)	873.00	b
$\mathcal{F}$	2014-2015	0.73 (0.23)	827.50	b	0.36 (0.17)	800.50	b	0.87 (0.06)	294.00	d
$\mathcal{E}$	2014-2015	0.66 (0.25)	637.50	c	0.26 (0.14)	533.50	c	0.94 (0.03)	943.00	a
$\mathcal{E}^D$	2014-2015	0.54 (0.26)	323.50	d	0.23 (0.12)	276.00	d	0.93 (0.03)	660.00	c
$\mathcal{A}^E$	2013	0.76 (0.20)	410.50	a	0.37 (0.14)	335.00	a	0.94 (0.02)	380.00	a
$\mathcal{F}$	2013	0.73 (0.21)	354.00	b	0.35 (0.15)	285.00	b	0.89 (0.04)	129.00	c
$\mathcal{E}$	2013	0.62 (0.23)	246.50	c	0.24 (0.11)	193.00	c	0.93 (0.03)	374.00	a
$\mathcal{E}^D$	2013	0.48 (0.24)	119.00	d	0.20 (0.11)	97.00	d	0.93 (0.03)	247.00	b

$\mathcal{A}^W$ and $\mathcal{W}^D$ in terms of CP_k. In particular, we measure the performance loss due to concept drift as the difference between the CP_k in the first and the second month, divided by the value of CP_k in the first month. Our experiments show that on these 10 datasets the CP_k of $\mathcal{A}^W$ decays of 7.7%, while the CP_k of $\mathcal{W}^D$ decays of 12.5%, confirming the superior adaptation performance of the proposed learning strategy.

E. Sample Selection Bias Due to Alert-Feedback Interaction

We here investigate whether importance weighting [19], a mainstream solution to correct SSB, can successfully compensate the SSB introduced by the alert-feedback interaction. To this purpose, we consider the feedback classifier $\mathcal{F}_t$, as this is the one mainly affected by the SSB due to alert-feedback interaction, and use a weight-sensitive implementation of the Random Forests based on conditional inference trees [40].

Importance weighting [32], [67], [68] consists of re-weighting each training sample in F_t according to the weight w defined as:

$$w = \frac{\mathcal{P}(\mathbf{s}=1)}{\mathcal{P}(\mathbf{s}=1|x,y)}, \tag{10}$$

where $\mathbf{s}$ is a selection variable that associates to each sample in T_t the value 1 if the transaction is in F_t and 0 otherwise, thus $\mathcal{P}(\mathbf{s}=1|x,y)$ corresponds to the probability for a sample (x,y) to be in the training set F_t. The definition of weights in (10) follows from the Bayes theorem and the fact that it is possible to express (as in [19]) the unbiased join distribution $\mathcal{P}(x,y)$ w.r.t. the biased joint distribution $\mathcal{P}(x,y|\mathbf{s}=1)$ as

$$\mathcal{P}(x,y) = \frac{\mathcal{P}(\mathbf{s}=1)}{\mathcal{P}(\mathbf{s}=1|x,y)}\mathcal{P}(x,y|\mathbf{s}=1) = w\mathcal{P}(x,y|\mathbf{s}=1).$$

Table IV reports the performance achieved when correcting the SSB using weights provided by (10) and it emerges that these are lower than the performance achieved by $\mathcal{F}$ in Table III. Thus, importance weighting does not actually improve the performance of $\mathcal{F}$ as it cannot compensate the SSB introduced by the alert-feedback interaction.

We believe that importance weighting turns to be ineffective since $\mathcal{P}(\mathbf{s}=1|x,y)$ and $\mathcal{P}(+|x)$ in (10) are highly correlated, due to the alert-feedback interaction. It means that, the more a transaction is likely to be considered risky, the greater the probability $\mathcal{P}(\mathbf{s}=1|x,y)$ and the lower is its weight in (10). This means that importance weighting techniques lower the influence of those samples within the feedbacks that are more likely to be a fraud. This has a negative impact in terms of the alert precision.

TABLE IV
AVERAGE P_k, CP_k AND AUC FOR $\mathcal{F}_t$ WHEN $Q = 15$.

metric	mean	sd	dataset
P_k	0.68	0.26	2014-2015
P_k	0.59	0.26	2013
CP_k	0.26	0.16	2014-2015
CP_k	0.25	0.13	2013
AUC	0.85	0.06	2014-2015
AUC	0.85	0.06	2013

As a sanity check, we repeated this experiment in a framework where recent supervised samples are not provided by the alert-feedback interaction but are randomly selected (in the same number and class proportions of the above experiment) among transactions having amount larger than €500. This form of SSB is referred to as covariate shift [42], [58], [66], since we have $\mathcal{P}(\mathbf{s}|y,x) = \mathcal{P}(\mathbf{s}|x)$, i.e. the selection variable $\mathbf{s}$ is independent of the class y given the input x. In this case, importance weighting was able to correctly compensate for this bias, and the de-biased clasifier outperforms a similar classiifer trained without correcting the SSB.

F. Influence of Parameters

We here show how the performance of $\mathcal{F}_t$ and $\mathcal{A}_t^W$ are influenced by: *i*) the number of feedback days considered to train our classifiers everyday (i.e., Q) and, *ii*) the number of number of cards to be investigated everyday. To this purpose we consider $\delta = 15$ days of verification latency, such that $\mathcal{F}_t$ is trained on 30 days of feedbacks ($Q = 30$, $M = 15$, $\delta = 15$) and the delayed supervised samples come only after 15 days. In Table V we see that, with $Q = 30$, $\mathcal{F}$ achieves superior performance in terms of CP_k when it is trained with more

feedbacks $Q = 15$ (see Table III). The same holds for $\mathcal{A}^W$, as a consequence of the superior performance achieved by $\mathcal{F}$. Therefore, in this case, the larger amount of feedbacks used during training well compensate the increase of verification latency.

TABLE V
AVERAGE CP_k WHEN USING 30 DAYS OF TRANSACTIONS ($\delta = 15$, $M = 15, Q = 30$).

classifier	mean	sd	sum of ranks	comparison	dataset
$\mathcal{A}^W$	0.38	0.17	1671.00	a	2014-2015
$\mathcal{F}$	0.36	0.17	1482.50	b	2014-2015
$\mathcal{R}$	0.31	0.17	1234.50	c	2014-2015
$\mathcal{W}$	0.25	0.13	850.50	d	2014-2015
$\mathcal{W}^D$	0.24	0.12	705.50	e	2014-2015
$\mathcal{S}$	0.23	0.12	605.50	f	2014-2015
$\mathcal{A}^W$	0.38	0.14	609.00	a	2013
$\mathcal{F}$	0.35	0.14	541.00	b	2013
$\mathcal{R}$	0.27	0.11	411.50	c	2013
$\mathcal{W}$	0.25	0.13	325.50	d	2013
$\mathcal{W}^D$	0.24	0.12	281.00	e	2013
$\mathcal{S}$	0.20	0.12	198.00	f	2013

We repeat this experiment by considering a larger number of feedbacks per day, to show how this parameter influences the performance of $\mathcal{F}$ and $\mathcal{A}^W$. In Table VI we assume that investigators are able to check more than 100 cards, and report the fraud detection performance in terms of NCP_k to properly assess alert precision when more cards can be controlled. This result confirms that having more feedbacks guarantees superior fraud detection performance. This analysis can be considered as a guideline for companies that have to decide whether the costs of hiring more investigators is compensated by the expected improvement in the fraud detection performance.

Another important parameter in our learning strategy is α, which balances the contribution of the feedback and delayed classifiers in (9). This was empirically set to 0.5 after having investigated multiple strategies to make this parameter adaptive on a daily basis. Our idea was to take into account the precision (or other performance measures) achieved during day $t-1$ by the $\mathcal{F}_{t-1}$ and $\mathcal{D}_{t-1}$, and then assigning weights to $\mathcal{F}_t$ and $\mathcal{D}_t$ accordingly (the best the classifier was during day $t-1$, the larger the weight during day t). Unfortunately, none of the implemented solutions seemed to outperforms the average of the two posteriors, i.e., $\alpha_t = 0.5\,\forall t$.

Thus, we ran an extensive simulation on the sliding window solution, where we tested everyday $\alpha_t \in \{0.1, 0.2, \ldots, 0.9\}$ and then we choose α_t^* as the one yielding the aggregation performing at best in terms of P_k. Such an optimal selection of weights is of course not feasible in a real-world FDS, as would necessary require to request feedbacks for each $\alpha_t \in \{0.1, 0.2, \ldots, 0.9\}$. However, this experiment showed that setting everyday α_t^* provided minimal improvements with respect to setting $\alpha_t = 0.5\,\forall t$. This can be explained by the fact that α_t^* had a peaked distribution about 0.5, having $\text{mean}(\alpha_t^*) \approx 0.52$. The P_k value was steadily decreasing when approaching $\alpha = 0.1$ and $\alpha = 0.9$, indicating that extreme values of α are very seldom the best option. In these extreme cases, $\mathcal{A}_t$ approaches either $\mathcal{D}_t$ or $\mathcal{F}_t$ (which are shown not to be the best options) and the classifier receiving the lowest weight has little chances to request feedbacks in order to improve its performance and increase its weight.

TABLE VI
AVERAGE NCP_k WHEN $k \geq 100$ IN THE 2013 DATASET ($\delta = 15$).

classifier	mean	sd	sum of ranks	comparison	k
$\mathcal{A}^W$	0.48	0.09	506.00	a	300
$\mathcal{F}$	0.46	0.10	448.00	b	300
$\mathcal{W}$	0.38	0.11	283.00	c	300
$\mathcal{W}^D$	0.35	0.10	172.50	d	300
$\mathcal{A}^W$	0.41	0.10	519.50	a	150
$\mathcal{F}$	0.38	0.10	441.50	b	150
$\mathcal{W}$	0.29	0.10	272.50	c	150
$\mathcal{W}^D$	0.27	0.09	179.50	d	150
$\mathcal{A}^W$	0.40	0.13	518.50	a	100
$\mathcal{F}$	0.37	0.13	443.00	b	100
$\mathcal{R}$	0.29	0.10	342.50	c	100
$\mathcal{W}^D$	0.26	0.11	249.00	d	100

VII. CONCLUSIONS

The majority of works addressing the fraud detection problem in credit card transactions (e.g. [5], [23], [61]) unrealistically assumes that the class of each transaction is immediately provided for training the classifier. We here analyze in detail the real-world working conditions of FDS and provide a formal description of the articulated classification problem involved. In particular, we have described the alert-feedback interaction, which is the mechanism providing recent supervised samples to train/update the classifier. We also claim that, in contrast with traditional performance measures used in the literature, in a real-world FDS, the precision of the reported alerts is probably the most meaningful one, since investigators can check only few alerts.

Our experiments on two vast datasets of real-world transactions show that, in order to get precise alerts, it is mandatory to assign larger importance to feedbacks during the learning problem. Not surprisingly, feedbacks play a major role in the proposed learning strategy, which consists in separately training a classifier on feedbacks and one on delayed supervised samples. Our experiments also show that solutions which lower the influence of feedbacks in the learning process (e.g. classifiers that mix feedbacks and delayed supervised samples or that implement some instance weighting schemes) are often returning less precise alerts.

Future work concerns the study of adaptive and possibly non-linear aggregation methods for the classifiers trained on feedbacks and delayed supervised samples. We also expect to further increase the alert precision by implementing a *learning to rank* approach [46] that would be specifically designed to replace the linear aggregation of the posterior probabilities. Finally, a very promising research direction concerns semi-supervised learning methods [16] for exploiting in the learning process also few recent, unlabeled transactions.

ACKNOWLEDGMENT

A. Dal Pozzolo is supported by the *Doctiris* scholarship funded by Innoviris, Belgium. G. Bontempi is supported by the projects *BridgeIRIS* and *BruFence* funded by Innoviris, Belgium.

REFERENCES

[1] E. Aleskerov, B. Freisleben, and B. Rao. Cardwatch: A neural network based database mining system for credit card fraud detection. In *Computational Intelligence for Financial Engineering (CIFEr), 1997., Proceedings of the IEEE/IAFE 1997*, pages 220–226. IEEE, 1997.

[2] C. Alippi, G. Boracchi, and M. Roveri. A just-in-time adaptive classification system based on the intersection of confidence intervals rule. *Neural Networks*, 24(8):791–800, 2011.

[3] C. Alippi, G. Boracchi, and M. Roveri. Hierarchical change-detection tests. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–13, 2016.

[4] C. Alippi, G. Boracchi, and M. Roveri. Just-in-time classifiers for recurrent concepts. *Neural Networks and Learning Systems, IEEE Transactions on*, 24(4):620–634, April.

[5] B. Baesens, V. Van Vlasselaer, and W. Verbeke. *Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques: A Guide to Data Science for Fraud Detection*. John Wiley & Sons, 2015.

[6] A. C. Bahnsen, D. Aouada, and B. Ottersten. Example-dependent cost-sensitive decision trees. *Expert Systems with Applications*, 2015.

[7] A. C. Bahnsen, D. Aouada, A. Stojanovic, et al. Detecting credit card fraud using periodic features. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 208–213. IEEE, 2015.

[8] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland. Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3):602–613, 2011.

[9] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *SDM*, volume 7, page 2007. SIAM, 2007.

[10] R. Bolton and D. Hand. Statistical fraud detection: A review. *Statistical Science*, pages 235–249, 2002.

[11] R. J. Bolton and D. J. Hand. Unsupervised profiling methods for fraud detection. *Credit Scoring and Credit Control VII*, pages 235–255, 2001.

[12] R. Brause, T. Langsdorf, and M. Hepp. Neural data mining for credit card fraud detection. In *Tools with Artificial Intelligence, Proceedings*, pages 103–106. IEEE, 1999.

[13] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[14] M. Carminati, R. Caron, F. Maggi, I. Epifani, and S. Zanero. *BankSealer: An Online Banking Fraud Analysis and Decision Support System*, pages 380–394. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[15] P. Chan, W. Fan, A. Prodromidis, and S. Stolfo. Distributed data mining in credit card fraud detection. *Intelligent Systems and their Applications*, 14(6):67–74, 1999.

[16] O. Chapelle, B. Schölkopf, A. Zien, et al. Semi-supervised learning. 2006.

[17] N. Chawla, K. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research (JAIR)*, 16:321–357, 2002.

[18] S. Chen and H. He. Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach. *Evolving Systems*, 2(1):35–50, 2011.

[19] C. Cortes, M. Mohri, M. Riley, and A. Rostamizadeh. Sample selection bias correction theory. In *Algorithmic learning theory*, pages 38–53. Springer, 2008.

[20] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi. Credit card fraud detection and concept-drift adaptation with delayed supervised information. In *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015.

[21] A. Dal Pozzolo, O. Caelen, and G. Bontempi. When is undersampling effective in unbalanced classification tasks? In *Machine Learning and Knowledge Discovery in Databases*. Springer, 2015.

[22] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi. Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE Symposium on Computational Intelligence and Data Mining*. IEEE, 2015.

[23] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Systems with Applications*, 41(10):4915–4928, 2014.

[24] A. Dal Pozzolo, R. A. Johnson, O. Caelen, S. Waterschoot, N. V. Chawla, and G. Bontempi. Using hddt to avoid instances propagation in unbalanced and evolving data streams. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 588–594. IEEE, 2014.

[25] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[26] G. Ditzler and R. Polikar. Incremental learning of concept drift from streaming imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 25(10):2283–2301, 2013.

[27] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar. Learning in nonstationary environments: A survey. *Computational Intelligence Magazine, IEEE*, 10(4):12–25, 2015.

[28] J. Dorronsoro, F. Ginel, C. Sgnchez, and C. Cruz. Neural fraud detection in credit card operations. *Neural Networks*, 8(4):827–834, 1997.

[29] C. Elkan. The foundations of cost-sensitive learning. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 973–978. Citeseer, 2001.

[30] R. Elwell and R. Polikar. Incremental learning of concept drift in nonstationary environments. *Neural Networks, IEEE Transactions on*, 22(10):1517–1531, 2011.

[31] W. Fan and I. Davidson. On sample selection bias and its efficient correction via model averaging and unlabeled examples. In *SDM*, pages 320–331. SIAM, 2007.

[32] W. Fan, I. Davidson, B. Zadrozny, and P. S. Yu. An improved categorization of classifier's sensitivity on sample selection bias. In *Data Mining, Fifth IEEE International Conference on*, pages 4–pp. IEEE, 2005.

[33] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Advances in artificial intelligence–SBIA 2004*, pages 286–295. Springer, 2004.

[34] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):44, 2014.

[35] J. a. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, Mar. 2014.

[36] J. Gao, B. Ding, W. Fan, J. Han, and P. S. Yu. Classifying data streams with skewed class distributions and concept drifts. *Internet Computing*, 12(6):37–49, 2008.

[37] S. Ghosh and D. L. Reilly. Credit card fraud detection with a neural-network. In *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, volume 3, pages 621–630. IEEE, 1994.

[38] D. Hand. Measuring classifier performance: a coherent alternative to the area under the roc curve. *Machine learning*, 77(1):103–123, 2009.

[39] H. He and E. A. Garcia. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, 2009.

[40] T. Hothorn, P. Bühlmann, S. Dudoit, A. Molinaro, and M. J. Van Der Laan. Survival ensembles. *Biostatistics*, 7(3):355–373, 2006.

[41] S. Jha, M. Guillen, and J. C. Westland. Employing transaction aggregation strategy to detect credit card fraud. *Expert systems with applications*, 39(16):12650–12657, 2012.

[42] M. G. Kelly, D. J. Hand, and N. M. Adams. The impact of changing populations on classifier performance. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 367–371. ACM, 1999.

[43] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research*, 8:2755–2790, 2007.

[44] G. Krempl and V. Hofer. Classification in presence of drift and latency. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 596–603. IEEE, 2011.

[45] M. Krivko. A hybrid model for plastic card fraud detection systems. *Expert Systems with Applications*, 37(8):6070–6076, 2010.

[46] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[47] N. Mahmoudi and E. Duman. Detecting credit card fraud by modified fisher discriminant analysis. *Expert Systems with Applications*, 42(5):2510–2516, 2015.

[48] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.

[49] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530, 2012.

[50] K. Nishida and K. Yamauchi. Detecting concept drift using statistical testing. In *Discovery Science*, pages 264–269. Springer, 2007.

[51] D. Olszewski. Fraud detection using self-organizing map visualizing the user profiles. *Knowledge-Based Systems*, 70:324–334, 2014.

[52] C. Phua, V. Lee, K. Smith, and R. Gayler. A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*, 2010.

[53] J. T. Quah and M. Sriganesh. Real-time credit card fraud detection using computational intelligence. *Expert Systems with Applications*, 35(4):1721–1732, 2008.
[54] Y. Sahin, S. Bulkan, and E. Duman. A cost-sensitive decision tree approach for fraud detection. *Expert Systems with Applications*, 40(15):5916–5923, 2013.
[55] D. Sánchez, M. Vila, L. Cerda, and J. Serrano. Association rules applied to credit card fraud detection. *Expert Systems with Applications*, 36(2):3630–3640, 2009.
[56] J. C. Schlimmer and R. H. Granger Jr. Incremental learning from noisy data. *Machine learning*, 1(3):317–354, 1986.
[57] B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52:55–66, 2010.
[58] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.
[59] W. N. Street and Y. Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–382. ACM, 2001.
[60] D. K. Tasoulis, N. M. Adams, and D. J. Hand. Unsupervised clustering in streaming data. In *ICDM Workshops*, pages 638–642, 2006.
[61] V. Van Vlasselaer, C. Bravo, O. Caelen, T. Eliassi-Rad, L. Akoglu, M. Snoeck, and B. Baesens. Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems*, 2015.
[62] S. Wang, L. L. Minku, and X. Yao. Resampling-based ensemble methods for online class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1356–1368, May 2015.
[63] D. Weston, D. Hand, N. Adams, C. Whitrow, and P. Juszczak. Plastic card fraud detection using peer group analysis. *Advances in Data Analysis and Classification*, 2(1):45–62, 2008.
[64] C. Whitrow, D. J. Hand, P. Juszczak, D. Weston, and N. M. Adams. Transaction aggregation as a strategy for credit card fraud detection. *Data Mining and Knowledge Discovery*, 18(1):30–55, 2009.
[65] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.
[66] K. Yamazaki, M. Kawanabe, S. Watanabe, M. Sugiyama, and K.-R. Müller. Asymptotic bayesian generalization error when training and test distributions are different. In *Proceedings of the 24th international conference on Machine learning*, pages 1079–1086. ACM, 2007.
[67] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114. ACM, 2004.
[68] B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Data Mining, ICDM*, pages 435–442. IEEE, 2003.
[69] V. Zaslavsky and A. Strizhak. Credit card fraud detection using self-organizing maps. *Information and Security*, 18:48, 2006.
[70] I. Žliobaitė. Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*, 2010.

Andrea Dal Pozzolo received in 2011 a master degree *cum laude* from the faculty of Statistics, Universitá di Bologna (Italy) and later in 2015 a PhD from the Machine Learning Group of the Université Libre de Bruxelles (Belgium). His research focuses on Machine Learning techniques for Fraud Detection. In particular he is interested in techniques for unbalanced data streams, cost-sensitive learning and concept drift.

Giacomo Boracchi received the M.S. degree in Mathematics from the Universita Statale degli Studi di Milano, Italy, and the Ph.D. degree in Information Technology at Politecnico di Milano, Italy, in 2004 and 2008, respectively. He was researcher at Tampere International Center for Signal Processing, Finland, during 2004-2005. Currently, he is an assistant professor at the Dipartimento di Elettronica, Informazione e Bioingegneria of the Politecnico di Milano.

Olivier Caelen got his PhD at the Machine Learning Group of the Université Libre de Bruxelles under the supervision of Prof. Gianluca Bontempi. His research focused mainly on computational intelligence and in particular on methods to speed up the models selection process. He is also interested in the multi armed bandit problem. He currently works at Worldline SA (Belgium) in the R&D High Processing & Volume team.

Cesare Alippi (S92-M97-SM99-F06) received the degree (cum laude) in electronic engineering and the Ph.D. degree from the Politecnico di Milano, Milan, Italy, in 1990 and 1995, respectively. He is currently a Full Professor of information processing systems with the Politecnico di Milano. He has been a Visiting Researcher with University College London, U.K., Massachusetts Institute of Technology, United States, ESPCI, France, and CASIA, China. He has authored or co-authored about 200 papers in international journals and conference proceedings, and holds five patents. His current research interests include adaptation and learning in nonstationary environments and intelligent embedded systems. Dr. Alippi was a recipient of the IEEE Instrumentation and Measurement Society Young Engineer Award in 2004, the Knight of the Order of Merit of the Italian Republic in 2011. He is the Vice-President Education of the IEEE Computational Intelligence Society and an Associate Editor of the IEEE Computational Intelligence Magazine. He was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS, the IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENTS from 2003 to 2009, and a member and the Chair of other IEEE committees.

Gianluca Bontempi graduated with honors in Electronic Engineering (Politecnico of Milan, Italy) and obtained his PhD in Applied Sciences (ULB, Brussels, Belgium). He took part to research projects in academy and private companies all over Europe. His interests cover data mining, machine learning, bioinformatics, time series prediction and simulation. He is author of more than 100 scientific publications and IEEE Senior Member. He is also co-author of software for data mining and prediction, which was awarded in two international competitions. Since January 2002 he is Professor in the Computer Science Department of ULB and Head of the ULB Machine Learning Group. In 2013 he has been appointed Director of the Interuniversity Institute of Bioinformatics in Brussels $(\text{IB})^2$.