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

Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, *Fellow, IEEE*, and Gianluca Bontempi, *Senior Member, IEEE*

Abstract—Detecting frauds in credit card transactions is perhaps one of the best testbeds 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 rely on assumptions that hardly hold in a real-world fraud-detection system (FDS). This lack of realism concerns two main aspects: 1) the way and timing with which supervised information is provided and 2) the measures used to assess fraud-detection performance. This paper has three major contributions. First, we propose, with the help of our industrial partner, a formalization of the fraud-detection problem that realistically describes the operating conditions of FDSs that everyday analyze massive streams 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 novel learning strategy that effectively addresses class imbalance, concept drift, and verification latency. Third, in our experiments, we demonstrate the impact of class unbalance and concept drift in a real-world data stream containing more than 75 million transactions, authorized over a time window of three years.

Index Terms—Concept drift, Credit card fraud detection, learning in nonstationary environments, unbalanced classification.

I. INTRODUCTION

CREDIT card fraud detection is a relevant problem that draws the attention of machine-learning and computational intelligence communities, where a large number of

automatic solutions have been proposed [1], [6], [8], [23], [24], [41], [47], [55], [56], [66]. In fact, this problem appears to be particularly challenging from a learning perspective, 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. 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 each alerted 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, in order 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 nondisputed 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 [53] 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 [6], [8], [12], [23], [24], [28], [47], [55]). 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 Fraud-Detection System (FDS) concerns the measures used to assess the fraud-detection performance. Most often, global ranking measures [23], [24], [63], like the area under the
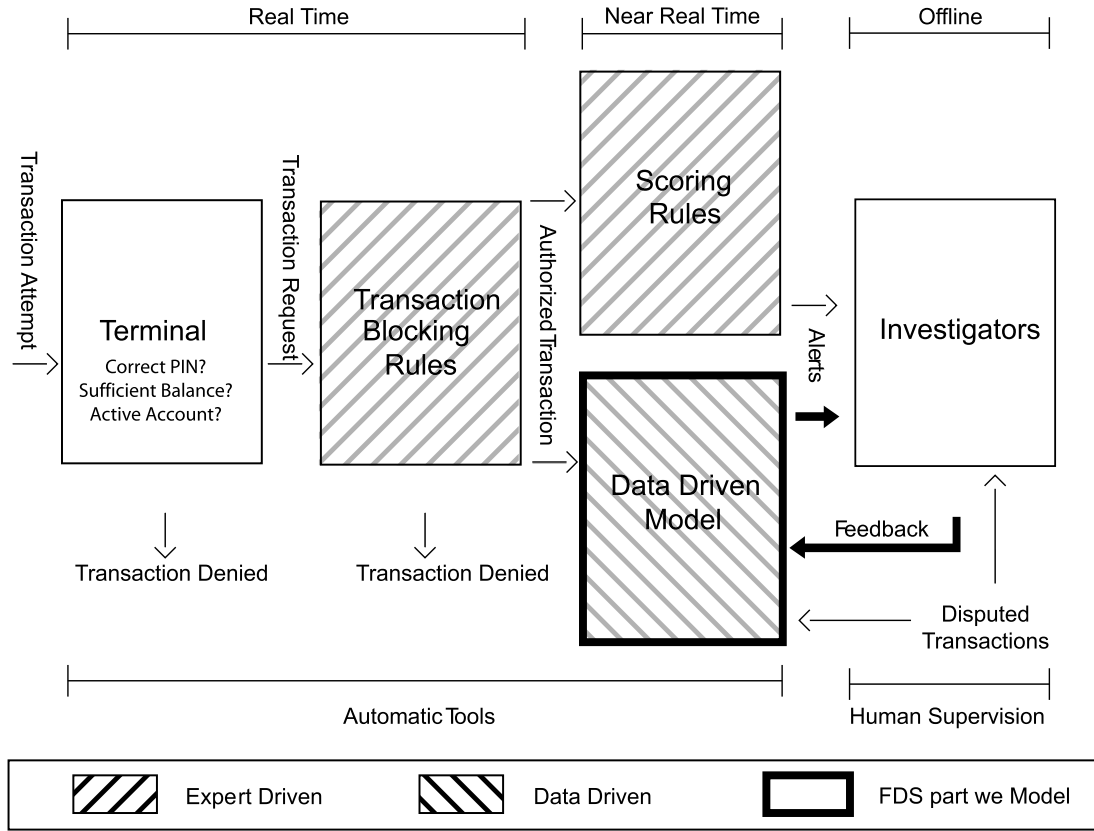
Fig. 1. Scheme illustrating the layers of control in an FDS. Our focus is mainly on the DDM and the *alert–feedback* interaction, which regulates the way recent supervised samples are provided.

ROC curve (AUC), or cost-based measures [6], [47], [55] are used, but these ignore the fact 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 as follows.

1) 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.
2) We introduce the performance measures that are considered in a real-world FDS.
3) Within this sound and realistic model, we propose an effective learning strategy for addressing the above challenges, including the verification latency and the alert–feedback interaction. This learning strategy is tested on a large number of credit card transactions.

This paper is organized as follows. We first detail the operating conditions of a real-world FDS in Section II, and then in Section III 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 (or 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 [20], which is significantly extended by describing in detail the real-world operating conditions of an 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 data sets.

II. REAL-WORLD FDS

Here we describe the main peculiarities and the operating conditions of a real-world FDS, inspired by the one routinely used by our industrial partner. Fig. 1 illustrates the five layers of control typically employed in an FDS: 1) the terminal; 2) the transaction-blocking rules; 3) the scoring rules; 4) the data-driven model (DDM); and 5) the investigators.

Layers 1)–4) fully implement automatic controls, while layer 5) is the only one requiring human intervention.

A. Layers of Controls in an FDS

1) Terminal: The terminal represents the first control layer in an FDS and performs conventional security checks on all the payment requests [63]. Security checks include controlling the

PIN code (possible only in case of cards provided with chip), the number of attempts, the card status (either active or blocked), the balance available, and the expenditure limit. In case of online transactions, these operations have to be performed in real time (response has to be provided in a few milliseconds), during which the terminal queries a server of the card issuing company. 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 are clearly perceived as 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 unsecured Web site 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 investigator and, as such, are *expert-driven* components of the FDS. To guarantee real-time operations and avoid blocking many genuine transactions, blocking rules should be: 1) quick to compute and 2) very precise, namely, should raise very few false alarms.

All transactions passing blocking rules are finally 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 1 h from the previous transaction THEN fraud score = 0.95.*"

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 another statistical model to estimate the probability for each feature vector being a fraud.

This probability is used as the fraud score associated with the authorized transactions. Thus, the DDM is trained from a set of labeled transactions and cannot be interpreted or manually modified by investigators. An effective DDM 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 fraud-detection performance. Transactions associated with feature vectors that have either received a large fraud score or a high probability of being 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 experienced in analyzing credit card transactions and are responsible of the expert-driven layers of the FDS. In particular, investigators design 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 is reported, including the assigned scores/probabilities, which in practice indicate how *risky* each transaction is. Investigators call cardholders and, after having verified, assign the label "genuine" or "fraudulent" to the alerted transaction, and return this information to the FDS. In the following, we refer to these labeled transactions as *feedbacks* and use the term *alert–feedback interaction* to describe this mechanism yielding 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, not necessarily corresponding 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 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, cardholder ID, purchase amount, date, and time. All transaction requests passing the blocking rules are entered in a database containing all recent authorized transactions, where the feature-augmentation process starts. During feature augmentation, a specific set of *aggregated* features associated with each authorized transactions is 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, and the location of the last purchases [7], [8], [23], [41],

---

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

[45], [66]. Van Vlasselaer *et al.* [63] show that additional informative features can be extracted from the social networks connecting the cardholders with 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 unusual compared with the shopping habits of the 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], [63]. 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 might introduce substantial delays.

All 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 number of days have passed without cardholder dispute, all the unreported transactions are considered genuine by default, and inserted in the training set of the DDM.

Overall, there are two types of supervised information: 1) feedbacks provided by investigators that are limited in number but refer to recent transactions and 2) 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 nondisputed transactions.

D. System Update

Customers' spending behavior evolves and fraudsters continuously design new attacks, and 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 cannot modify the DDM, since it is not interpretable and can be only updated (e.g., retrained) on the basis of recent supervised information, as shown in Fig. 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 are available.

III. PROBLEM FORMULATION

Here, we model the classification problem to be addressed in a real-world FDS, providing a formal description of the alert–feedback interaction and presenting suitable performance measures. The proposed learning strategy (Section V) and our experiments (Section VI) are built upon this model.

Let x_i denote the feature vector associated with the ith 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

$$A_t = \{x_i \in T_t \text{ s.t. } r(x_i) \leq k\} \quad (1)$$

where $r(x_i) \in \{1, \ldots, |T_t|\}$ 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.[2] As discussed in Section II-A.5, investigators contact the cardholders and provide supervised samples to the FDS in the form of *feedbacks*. In particular, feedbacks include all recent transactions from the controlled cards, which we model as

$$F_t = \{(x_i, y_i) \text{ s.t. } x_i \text{ is from cards}(A_t)\} \quad (2)$$

where cards(A_t) denotes the set of cards having at least a transaction in A_t. The number of feedbacks, i.e., $|F_t|$, depends on the number of transactions associated with the k controlled cards.

After a certain verification latency, the labels of all the transactions are provided to the FDS, since, as discussed in Section II-C, nondisputed 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*

$$D_{t-\delta} = \{(x_i, y_i), \quad x_i \in T_{t-\delta}\}. \quad (3)$$

Note that $F_{t-\delta} \subset D_{t-\delta}$ since transactions at day $t-\delta$ obviously include those that have been alerted. Fig. 2 illustrates the different types of supervised information available in an 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. We further comment that in principle, since the classifier

---

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

3788 IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, VOL. 29, NO. 8, AUGUST 2018
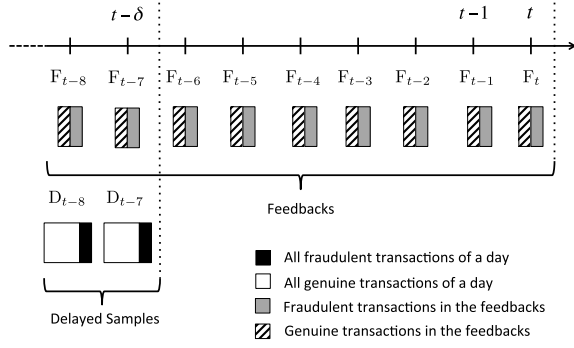

Fig. 2. Supervised samples available at the end of day t include: 1) feedbacks $[F_{(\cdot)}]$ and 2) delayed couples $[D_{(\cdot)}]$ occurred before $t - \delta$ days. In this plot, we have assumed $\delta = 7$. Patterns indicate different labels, and the size of these regions indicates balanced/unbalanced class proportions.

analyzes each feature vector x_i independently, it does not alert cards receiving several risky transactions until any of these enters in the pool of the alerts (1). However, these situations are particularly relevant for investigators, and can be handled either by suitable scoring rules or feature augmentation, adding, for instance, a component that keeps track of the scores of recent transactions.

Fraud-detection performance can be conveniently assessed in terms of the *alert precision* $P_k(t)$, which is defined as

$$P_k(t) = \frac{|TP_k(t)|}{k} \quad (4)$$

where $TP_k(t) = \{(x_i, y_i) \text{ such that } x_i \in A_t, y_i = +\}$. Thus, $P_k(t)$ is the proportion of frauds in the alerts A_t. Though the classifier independently processes 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

$$CP_k(t) = \frac{|C_t^+|}{k} \quad (5)$$

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* $CP_k(t)$ as

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

where $\Gamma(t)$ is the maximum value of $CP_k(t)$ and γ_t is the number of fraudulent cards at day t. From (6), we have that $NCP_k(t)$ takes values in the range $[0, 1]$, while $CP_k(t)$ is in $[0, 1]$ when $\gamma_t > k$ and in $[0, (\gamma_t/k)]$ otherwise. For example,

if at day t, we have correctly detected 40 fraudulent cards ($|C_t^+| = 40$) out of the $k = 100$ cards checked by investigators, and the overall number of fraudulent cards is 50 ($\gamma_t = 50$), then $CP_k(t) = 0.4$ while $NCP_k(t) = (0.4)/(0.5) = 0.8$.

Note that, since $\Gamma(t)$ does not depend on the specific classifier $\mathcal{K}_{t-1}$ adopted, when algorithm "A" is better than algorithm "B" in terms of CP_k, "A" is also better than "B" in terms of NCP_k. Moreover, because of verification latency, the number of fraudulent cards in day t (i.e., γ_t) can be only computed after few days, and therefore NCP_k cannot be computed in real time. Thus, we recommend using CP_k for assessing the 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], [62] 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 an FDS can be directly configured from unlabeled transactions. A well-known method is peer group analysis [65], which clusters customers according to their profile and identifies frauds as transactions departing from the typical cardholder's behavior (also see [52]). The typical cardholder's behavior has also been modeled by means of self-organizing maps [51], [54], [71].

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, including neural networks [1], [12], [28], logistic regression [41], association rules [56], support vector machines [66], modified Fisher discriminant analysis [47], and decision trees [6], [24], [55]. Several studies have reported random forest (RF) to achieve the best performance [8], [20], [23], [63], [66]: this is one of the reasons why we adopt RFs in our experiments.

B. Performance Measure for Fraud Detection

The typical performance measure for fraud-detection problems is the AUC [23], [24], [63]. 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 [37]. Another ranking measure frequently used in fraud detection is average precision [23], which corresponds to the area under the precision–recall curve. While these measures are widely used in detection problems, cost-based measures have been specifically designed for fraud-detection purposes. Cost-based measures [6], [47], [55] quantify the monetary loss of a fraud by means of a cost matrix that associates a cost with each entry of the confusion matrix. Elkan [29] shows that a cost matrix may be misleading because the minimum/maximum loss of

Authorized licensed use limited to: Bursa Uludag Universitesi. Downloaded on May 13,2023 at 15:30:56 UTC from IEEE Xplore. Restrictions apply.

the problem can change over time. To avoid this problem, *normalized* cost [66] or savings [6] are used to assess the performance with respect to 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, and thus an effective FDS should provide investigators a small number of reliable alerts. This is the reason why 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 an FDS include: 1) handling the class *imbalance*, since legitimate transactions far outnumber the fraudulent ones; 2) handling the *concept drift* since the statistical properties of both frauds and genuine transactions evolve with time; and 3) 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 [45] and [24] and in our analysis (see Table I). Learning under class imbalance has lately received a lot of attention, since 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 [38]. The two main approaches for dealing with class imbalance are: 1) *sampling* methods and 2) *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, which balance the class proportions in the training set by removing samples from the majority class, and oversampling ones, 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 samples belonging to the minority and majority class. In credit card fraud detection, the cost of a missed fraud is often assumed to be proportional to the transaction amount [6], [47], [55], 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], [35]. 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, 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 data sets 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 into two families: 1) active adaptation and 2) passive adaptation.

Active approaches [4], [9], [34], [50], [60] 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 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], [61], [72] and classifiers trained over a sliding window of the recent supervised samples (like STAGGER [57] and FLORA [67]) 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], [64]. An alternative approach consists in propagating the training samples of the minority class over time [36], possibly undersampling the majority class. Chen and He [18] proposed REA that propagates examples only 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 (see [11], [12], [15]) are tested in experiments where transaction labels are supposed to be available the very next day since the transaction is authorized. In a real-world FDS (Section II-C), the only recent supervised information is the feedbacks F_t, provided by investigators, while the vast majority of transactions authorized everyday do not receive a label in a short time ($|F_t| << |T_t|$). Feedbacks are not representative of the transactions processed everyday for two main reasons: 1) feedbacks contain transactions that are characterized by a high probability of being frauds and 2) 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 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 Sample Selection Bias (SSB). Here we simply mention that there are three different types of **SSBs!** (**SSBs!**): class-prior bias, feature bias (also called covariate shift), and complete bias. A standard remedy to SSB is *importance weighting* [32], [69], [70], namely, semisupervised reweighting techniques that assign larger weights to those training samples that more closely resemble 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 also been proposed to correct SSB [31].

The interaction between the FDS (raising alerts) and the investigators (providing true labels) recalls the active learning scenario [58], where it is possible 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. Considering the limited number of transactions investigators can check, addressing these questions would necessarily imply that some high-risk transaction is not being controlled, with the consequent loss in detection performance.

V. Proposed Learning Strategy

It is important to stress that feedbacks (F_t) and delayed samples ($D_{t-\delta}$) are very different sets of supervised samples. The first difference is quite evident: F_t provides recent up-to-date information while $D_{t-\delta}$ 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 $D_{t-\delta}$: while the class proportion in $D_{t-\delta}$ is heavily skewed toward 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 toward 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, and 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 $\{F_t, \ldots, F_{t-(Q-1)}\}$, and M days of delayed supervised samples $\{D_{t-\delta}, \ldots, D_{t-(\delta+M-1)}\}$, and these latter obviously include the feedbacks received in the same days (i.e., $F_i \subset D_i$, $i \leq t - \delta$). Our learning strategy, which is detailed in Algorithm 1, consists in separately training a classifier $\mathcal{F}_t$ on feedbacks

$$\mathcal{F}_t = \text{TRAIN}(\{F_t, \ldots, F_{t-(Q-1)}\}) \qquad (7)$$

and a classifier on delayed supervised samples

$$\mathcal{D}_t = \text{TRAIN}(\{D_{t-\delta}, \ldots, D_{t-(\delta+M-1)}\}) \qquad (8)$$

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

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

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).

The parameters Q and M that, respectively, define how many days of feedbacks and delayed supervised samples are used for training our classifiers have to be defined considering the overall number of feedbacks and the percentage of frauds. The training set of $\mathcal{F}_t$ approximately contains $Q \cdot |F_t|$ samples (a different number of feedbacks might be provided everyday) and this has to be a sufficiently large number to train a classifier addressing quite a challenging classification problem in high dimensions. However, Q cannot be made arbitrarily large, not to include old feedbacks. Similar considerations hold when setting M, the considered number of days containing delayed transactions, which have to include a sufficient number of frauds. Note that it is nevertheless possible to include in the training set of $\mathcal{F}_t$ feedbacks received before δ days ($Q \geq \delta$) and in particular in our experiments we used $Q = \delta + M$.

The rationale behind the proposed learning strategy is twofold. 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 correspond to values of $\mathcal{P}_{\mathcal{A}_t}$ that are very close to one. 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 that, however, might be obsolete because of concept drift.

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

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.

      $\mathcal{F}_{t+1} \leftarrow \mathrm{TRAIN}(\{\mathrm{F}_{t+1}, \dots, \mathrm{F}_{t-Q}\})$

      $\mathrm{D}_{t+1-\delta} \leftarrow$ transactions authorized at $t+1-\delta$

      $\mathcal{D}_{t+1} \leftarrow \mathrm{TRAIN}(\{\mathrm{D}_{t+1-\delta}, \dots, \mathrm{D}_{t-(\delta+M)}\})$

   return $\mathcal{F}_t, \mathcal{D}_t$ and $\mathcal{A}_t$ defined as in (9).

TABLE I

DATA SETS

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%

a *sliding window* classifier as in [62] and [63], which we denote by $\mathcal{W}_t^D$, while in the latter, $\mathcal{D}_t$ is an *ensemble* of classifiers similar to [36] and [23], 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 $\{\mathrm{D}_{t-\delta}, \dots, \mathrm{D}_{t-(\delta+M-1)}\}$. However, while $\mathcal{W}_t^D$ employs a unique model to this purpose, $\mathcal{E}_t^D$ is an ensemble of M classifiers $\{\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_M\}$ where each individual classifier $\mathcal{M}_i$ is trained on delayed samples of a different day, i.e., $\mathrm{D}_{t-\delta-i}, i = 0, \dots, M-1$. The posterior $\mathcal{P}_{\mathcal{E}_t^D}(+|x)$ is obtained by averaging the posterior probabilities of the individual classifiers, i.e., $\mathcal{P}_{\mathcal{E}_t^D}(+|x) = (\sum_i^M \mathcal{P}_{\mathcal{M}_i}(+|x))/(M)$.

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): $\{\mathrm{F}_t, \dots, \mathrm{F}_{t-(\delta-1)}, \mathrm{D}_{t-\delta}, \dots, \mathrm{D}_{t-(\delta+M-1)}\}$.

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 $\{\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_M, \mathcal{F}_t\}$, and whose posterior $\mathcal{P}_{\mathcal{E}_t}(+|x)$ is estimated by averaging the posterior probabilities of all its individuals, i.e., $\mathcal{P}_{\mathcal{E}_t}(+|x) = (\sum_i^M \mathcal{P}_{\mathcal{M}_i}(+|x) + \mathcal{P}_{\mathcal{F}_t}(+|x))/(M+1)$.

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., $\mathcal{F}_t, \mathcal{W}_t^D, \mathcal{W}_t, \mathcal{M}_i, i = 1, \dots, M$), we adopt 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 data set, 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 data sets and in Section VI-B, we detail the experimental settings. Section VI-C presents our first experiment that uses the classifiers described 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 Data Sets

We use two large data set of online 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 Fig. 1. In Table I, we provide all the information about these data sets, which we denote as 2013 and 2014–2015, and in particular, we stress the extreme class imbalance since frauds account for about 0.2% of all transactions. As shown in Fig. 3, the number of frauds per day varies significantly over time, and there are more fraudulent transactions than fraudulent cards, indicating that sometimes multiple frauds are perpetrated on the same card. The 2013 data set has also been used in [20] and part of this data set have been suitably anonymized and made publicly available for download [22].

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 data set 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
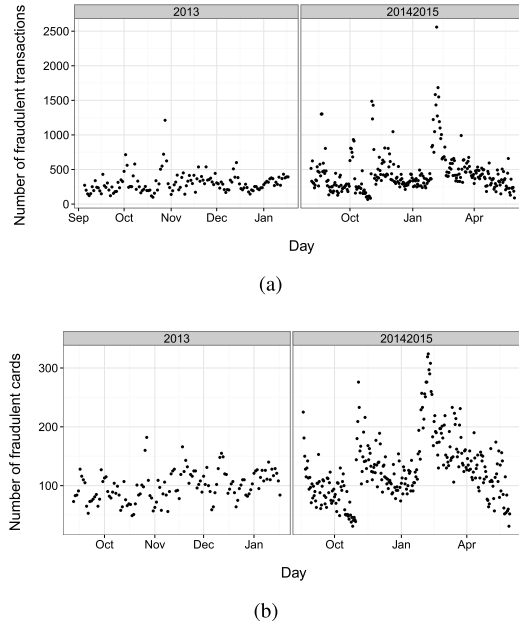
Fig. 3. Number of fraudulent transactions and cards per day in the data sets described in Table I. It emerges that there are more fraudulent transactions than cards, meaning that some cards have received more than a fraud. (a) Number of fraudulent transactions. (b) Number of fraudulent cards.
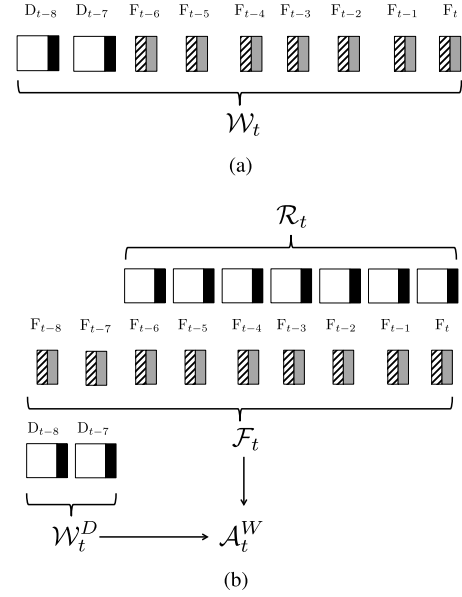

Fig. 4. Supervised information used by the classifiers considered in our experiments. In this illustrative example, we set $\delta = 7$, $M = 2$ and $Q = 7 + 2 = 9$. (a) Pooling together all labeled transactions. (b) Separating feedbacks and delayed samples.

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 is immediately blocked. A different option would be to remove all transactions of the same card after having detected the first fraud. However, this would reduce the number of available frauds, further worsening the class imbalance in our data set. Therefore, we consider the *CARD_ID* exclusively to compute the aggregated features, and do not include it in the feature vectors.

B. Experimental Settings

In agreement with our industrial partner, we assumed that investigators can check up to 100 cards alerted by the DDM every day. Thus, $\mathcal{F}_t$ is everyday trained over Q days containing each alerted transactions from 100 different cardholders. 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 also depend on the posterior of $\mathcal{D}_t$, while in the latter, alerts are uniquely determined by $\mathcal{F}_t$.

We assess the overall fraud-detection performance in our data sets both by averaging daily performance measures (P_k, CP_k, and AUC) and also by analyzing the sum of classifiers' ranks in each day. In particular, in each day j, we rank the S tested classifiers from the best to the least performing one, and denote by $r_{\mathcal{K},j} \in \{1, \dots, S\}$ the rank of the classifier $\mathcal{K}$ on day j: when $\mathcal{K}$ is the best classifier, its rank is maximum, i.e., $r_{\mathcal{K},j} = S$, while when it is the worst, $r_{\mathcal{K},j} = 1$. As recommended in [25], we perform a Friedman test [33] and reject the null hypothesis that all the

classifiers achieve the same performance. Then, we define a global ranking by summing all the daily ranks (see Table II): the larger the sum of ranks, the better the classifier, and we use paired t-tests to determine whether the differences in the global ranking are significant. In practice, for each pair of classifiers $\mathcal{K}$ and $\mathcal{H}$, a t-test is used to compare their ranks over all days (i.e., $r_{\mathcal{K},j} - r_{\mathcal{H},j}$, $j \in \{1, \dots, J\}$), being J the number of days.[3]

Each experiment is repeated 10 times to reduce performance's variability, and when comparing classifiers on multiple days, we omit the index t from the classifier notation. In most of our experiments, we consider one week of verification latency ($\delta = 7$) and $M = 8$, such that the overall number of feedbacks used is $Q = M + \delta = 15$. In Section VI-F, we repeat the experiments considering a longer verification latency $\delta = 15$ and $M = 15$, $Q = 30$.

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. $\mathcal{E}^D$). Fig. 4 illustrates the training set involved when using $\mathcal{A}_t^W$ and the related classifiers, while Table III summarizes the most important parameters and the training samples used by the considered classifiers.

In this experiment, we have also included the *ideal* classifier $\mathcal{R}_t$ that is trained on all transactions authorized between day t and $t-\delta$. This classifier is considered an ideal counterpart

---

[3]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 the same happens during the next few days. However, this is a standard *post hoc* analysis for nonparametric tests, such as the Friedman test [25].

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

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 (0.04)	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

TABLE III
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	$Q + M$

of sliding window classifiers, which unrealistically assume 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 II shows the average P_k, CP_k, and AUC over all the batches for the two data sets 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 data sets, $\mathcal{A}^W$ 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. In contrast, 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 note that CP_k is typically lower than P_k, since often multiple frauds are perpetrated on the same card.

Table II also reports the results in terms of AUC, a global ranking measure that evaluates the classifier's posteriors over all the instances and not only in the top k (differently from CP_k and P_k). In terms of AUC, the ideal classifier $\mathcal{R}$ is significantly better than $\mathcal{A}^W$ and $\mathcal{F}$ is by far the worse, indicating that $\mathcal{F}$ is not effective when ranking all the transactions.

We interpret these results as follows: when the goal is to obtain an accurate ranking of the most suspicious cards (i.e., maximize CP_k), we should assign larger weights to those transactions that are as risky as those we want to predict, hence using $\mathcal{A}^W$. 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 area under the ROC curve (AUC) of $\mathcal{R}$. In Table II, we can also see that $\mathcal{R}$ outperforms $\mathcal{W}^D$ in terms of P_k, CP_k, and AUC. This result suggests that the stream of credit card transactions is nonstationary. In fact, both the training sets of $\mathcal{R}$ and $\mathcal{W}^D$ contain all transactions authorized in $\delta = 7$ and $M = 8$ consecutive days, respectively. Their major difference is that $\mathcal{R}$ is trained on the most recent transactions, while transactions in $\mathcal{W}^D$ come with a lag of δ days. The fact that $\mathcal{R}$ outperforms $\mathcal{W}^D$ indicates that the most recent transactions are more informative to detect frauds in the next days, and thus that the distribution of transaction is nonstationary.

The standard deviations of P_k and CP_k, reported in Table II, are rather high in particular if compared with those of the AUC. As we discussed in Section III, and as shown in Fig. 5, the values of CP_k (and P_k as well) are heavily influenced by the number of frauds occurring every day. Since this number heavily fluctuates over time (see Fig. 3), it is reasonable to expect such a large dispersion. We remark that the comparison between classifiers in Table II indicates that differences in terms of performance are always significant, despite such a large standard deviation. Note that the values of NCP_k (see Table IV) are less affected by such fluctuations.

D. Concept Drift

In this section, we first analyze the 2014–2015 data set that contains more than 54 million transaction authorized over 10 months and show that this stream is affected by

TABLE IV
AVERAGE NCP_k WHEN $k \geq 100$ IN THE 2013 DATA SET ($\delta = 15$)

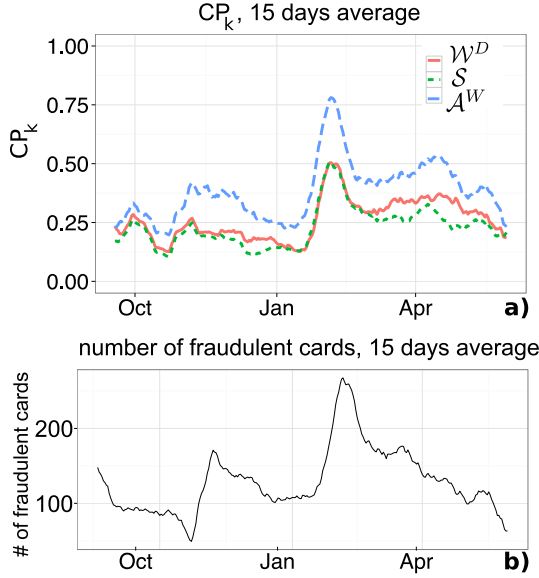| classifier	mean	sd	sum of ranks	comparison	k
$\mathcal{A}^W$	0.48	0.09	506.00	a	300
$\mathcal{F}$	0.46	0.10	448.00	b	300
$\mathcal{W}$	0.38	0.11	283.00	c	300
$\mathcal{W}^D$	0.35	0.10	172.50	d	300
$\mathcal{A}^W$	0.41	0.10	519.50	a	150
$\mathcal{F}$	0.38	0.10	441.50	b	150
$\mathcal{W}$	0.29	0.10	272.50	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

Fig. 5. (a) Values of CP_k for $\mathcal{S}$, $\mathcal{W}^D$, and $\mathcal{A}^W$ on data set 2014–2015. (b) Number of fraudulent cards on the same period. For the visualization sake, these values have averaged over a sliding window of 15 days. The peak of CP_k in (a) corresponds to the peak in number of fraudulent cards in (b). This result confirms that the classifiers become more precise in those days characterized by a large number of fraudulent cards.

concept drift. To this purpose, we use a static classifier $\mathcal{S}_t$, which is initially trained on M days and never updated (such that it initially coincides with $\mathcal{W}_t^D$), and compare it against $\mathcal{W}_t^D$ (which is instead regularly updated) and $\mathcal{A}_t^W$ (which also leverages updated supervised samples). In a stationary classification problem, the two classifiers $\mathcal{S}$ and $\mathcal{W}^D$ would perform similarly. The fact that $\mathcal{S}_t$ outperforms $\mathcal{W}_t^D$ over time [see Fig. 5(a)] confirms that this data set is affected by concept drift. While it might not sound surprising that the stream of credit card transactions is nonstationary, ours is, to the best of our knowledge, the first analysis on the impact of concept drift on such a large transaction data set.

Fig. 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 performances of all the classifiers in Fig. 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 data set [which is reported in Fig. 5(b)]. In contrast, during October 2014 (the period exhibiting the lowest number of

fraudulent cards in our data set), all the classifiers achieve low values of CP_k. Thus, Fig. 5 confirms that the alert precision heavily depends on the number of fraudulent cards in a day.

To further investigate the adaptation performance of $\mathcal{A}^W$ in nonstationary environments, we assess its adaptation abilities with respect to an artificially introduced concept drift. In particular, we artificially introduce changes at known locations, adding an abrupt drift on top of the (gradual) one affecting the transaction stream, which we have previously discussed. As in [20], we prepared 10 short streams by juxtaposing transactions authorized in two nonconsecutive months. Each of these streams contains an abrupt concept drift 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 $\mathcal{A}^W$ and $\mathcal{W}^D$ in terms of CP_k. In particular, we measure the relative 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 data sets 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

Here we investigate whether importance weighting [19], a mainstream solution to correct SSB, can successfully compensate for 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 RFs based on conditional inference trees [40].

Importance weighting [32], [69], [70] consists of reweighting each training sample in F_t using the following weight:

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

where $\mathbf{s}$ is a selection variable that associates with 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 $\mathcal{P}(x, y)$ with respect to 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 V 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 II. Importance weighting does not actually improve the performance of $\mathcal{F}$, which we interpret as a failure when compensating for 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

TABLE V
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

TABLE VI
AVERAGE CP_k WHEN USING 30 DAYS ($\delta = 15$, $M = 15$, AND $Q = 30$)

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

a transaction is likely to be considered risky, the larger the probability $\mathcal{P}(\mathbf{s} = 1|x, y)$ and the lower is its weight in (10), accordingly. Thus, importance weighting lowers the influence of those samples within the feedbacks that are more likely to be a fraud, and this negatively impacts on the alert precision.

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], [59], [68], since we have $\mathcal{P}(\mathbf{s}|y, x) = \mathcal{P}(\mathbf{s}|x)$, i.e., given the input x, the selection variable $\mathbf{s}$ is independent of the class y. In this case, importance weighting was able to correctly compensate for this bias, and the debiased classifier outperforms a similar classifier trained without correcting the SSB.

F. Influence of Parameters

Here we show how the performances of $\mathcal{F}_t$ and $\mathcal{A}_t^W$ are influenced by: 1) the number of feedback days considered to train our classifiers (i.e., Q); 2) the number of cards that are everyday controlled by investigators; and 3) the parameter α that regulates the aggregation classifier in (9). To this purpose, we consider $\delta = 15$ days of verification latency, such that $\mathcal{F}_t$ is trained on 30 days of feedbacks ($Q = 30$, $M = 15$, and $\delta = 15$) and the delayed supervised samples come after 15 days. Table VI shows that $\mathcal{F}$ is better in terms of CP_k when it is trained using $Q = 30$ days of feedbacks than $Q = 15$ (see Table II). The same holds for $\mathcal{A}^W$, as a consequence of the superior performance achieved by $\mathcal{F}$. Therefore, the larger amount of feedbacks used during training well compensate in this case the increase in verification latency.

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 IV, 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 the 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 are 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 assign 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 outperform 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 it would necessary require to request feedbacks for each $\alpha_t \in \{0.1, 0.2, \ldots, 0.9\}$. However, setting everyday α_t^* yielded 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.

VII. CONCLUSION

The majority of works addressing the fraud-detection problem in credit card transactions (see [5], [23], [63]) unrealistically assume that the class of each transaction is immediately provided for training the classifier. Here we 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 to 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 data sets 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 central role in the proposed learning strategy, which consists in separately

training a classifier on feedbacks and a classifier on delayed supervised samples, and then aggregating their posteriors to identify alerts. Our experiments also show that solutions that lower the influence of feedbacks in the learning process (e.g., classifiers that mix feedbacks and delayed supervised samples or that implement instance weighting schemes) are often returning less precise alerts.

Future work concerns the study of adaptive and possibly nonlinear 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], [39] for exploiting in the learning process also few recent unlabeled transactions.

REFERENCES

[1] E. Aleskerov, B. Freisleben, and B. Rao, "CARDWATCH: A neural network based database mining system for credit card fraud detection," in *Proc. IEEE/IAFE Computat. Intell. Financial Eng.*, Mar. 1997, pp. 220–226.

[2] C. Alippi, G. Boracchi, and M. Roveri, "A just-in-time adaptive classification system based on the intersection of confidence intervals rule," *Neural Netw.*, vol. 24, no. 8, pp. 791–800, 2011.

[3] C. Alippi, G. Boracchi, and M. Roveri, "Hierarchical change-detection tests," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 2, pp. 246–258, Feb. 2016.

[4] C. Alippi, G. Boracchi, and M. Roveri, "Just-in-time classifiers for recurrent concepts," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 4, pp. 620–634, Apr. 2013.

[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*. Hoboken, NJ, USA: Wiley, 2015.

[6] A. C. Bahnsen, D. Aouada, and B. Ottersten, "Example-dependent cost-sensitive decision trees," *Expert Syst. Appl.*, vol. 42, no. 19, pp. 6609–6619, 2015.

[7] A. C. Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, "Detecting credit card fraud using periodic features," in *Proc. 14th Int. Conf. Mach. Learn. Appl.*, Dec. 2015, pp. 208–213.

[8] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data mining for credit card fraud: A comparative study," *Decision Support Syst.*, vol. 50, no. 3, pp. 602–613, 2011.

[9] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proc. SDM*, vol. 7. 2007, pp. 443–448.

[10] R. Bolton and D. Hand, "Statistical fraud detection: A review," *Stat. Sci.*, vol. 17, no. 3, pp. 235–249, 2002.

[11] R. J. Bolton and D. J. Hand, "Unsupervised profiling methods for fraud detection," in *Credit Scoring Credit Control VII*. London, U.K.: Imperial College London, 2001, pp. 235–255.

[12] R. Brause, T. Langsdorf, and M. Hepp, "Neural data mining for credit card fraud detection," in *Proc. Tools Artif. Intell.*, Jul. 1999, pp. 103–106.

[13] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

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

[15] P. K. Chan, W. Fan, A. L. Prodromidis, and S. J. Stolfo, "Distributed data mining in credit card fraud detection," *IEEE Intell. Syst. Their Appl.*, vol. 14, no. 6, pp. 67–74, Nov. 1999.

[16] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning," *IEEE Trans. Neural Netw.*, vol. 20, no. 3, pp. 542–542, 2009.

[17] N. Chawla, K. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[18] S. Chen and H. He, "Towards incremental learning of nonstationary imbalanced data stream: A multiple selectively recursive approach," *Evolving Syst.*, vol. 2, no. 1, pp. 35–50, 2011.

[19] C. Cortes, M. Mohri, M. Riley, and A. Rostamizadeh, "Sample selection bias correction theory," in *Algorithmic Learning Theory*. Berlin, Germany: Springer, 2008, pp. 38–53.

[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 *Proc. Int. Joint Conf. Neural Netw.*, 2015, pp. 1–8.

[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*. Cambridge, U.K.: Springer, 2015.

[22] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, "Calibrating probability with undersampling for unbalanced classification," in *Proc. IEEE Symp. Ser. Computat. Intell.*, Dec. 2015, pp. 159–166.

[23] A. Dal Pozzolo, O. Caelen, Y.-A. L. Borgne, S. Waterschoot, and G. Bontempi, "Learned lessons in credit card fraud detection from a practitioner perspective," *Expert Syst. Appl.*, vol. 41, no. 10, pp. 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 *Proc. Int. Joint Conf. Neural Netw.*, 2014, pp. 588–594.

[25] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[26] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2283–2301, Oct. 2013.

[27] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Comput. Intell. Mag.*, vol. 10, no. 4, pp. 12–25, Apr. 2015.

[28] J. R. Dorronsoro, F. Ginel, C. Sgnchez, and C. S. Cruz, "Neural fraud detection in credit card operations," *IEEE Trans. Neural Netw.*, vol. 8, no. 4, pp. 827–834, Jul. 1997.

[29] C. Elkan, "The foundations of cost-sensitive learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2001, vol. 17. no. 1, pp. 973–978.

[30] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *Trans. Neural Netw.*, vol. 22, no. 10, pp. 1517–1531, 2011.

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

[32] W. Fan, I. Davidson, B. Zadrozny, and P. S. Yu, "An improved categorization of classifier's sensitivity on sample selection bias," in *Proc. 5th Int. Conf. Data Mining*, Nov. 2005, p. 4.

[33] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *J. Amer. Stat. Assoc.*, vol. 32, no. 200, pp. 675–701, 1937.

[34] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence*. Berlin, Germany: Springer, 2004, pp. 286–295.

[35] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, p. 44, 2014.

[36] J. Gao, B. Ding, W. Fan, J. Han, and P. S. Yu, "Classifying data streams with skewed class distributions and concept drifts," *IEEE Internet Comput.*, vol. 12, no. 6, pp. 37–49, Nov. 2008.

[37] D. Hand, "Measuring classifier performance: A coherent alternative to the area under the ROC curve," *Mach. Learn.*, vol. 77, no. 1, pp. 103–123, 2009.

[38] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[39] M. J. Hosseini, A. Gholipour, and H. Beigy, "An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams," *Knowl. Inf. Syst.*, vol. 46, no. 3, pp. 567–597, 2016.

[40] T. Hothorn, P. Bühlmann, S. Dudoit, A. Molinaro, and M. J. van der Laan, "Survival ensembles," *Biostatistics*, vol. 7, no. 3, pp. 355–373, 2006.

[41] S. Jha, M. Guillen, and J. C. Westland, "Employing transaction aggregation strategy to detect credit card fraud," *Expert Syst. Appl.*, vol. 39, no. 16, pp. 12650–12657, 2012.

[42] M. G. Kelly, D. J. Hand, and N. M. Adams, "The impact of changing populations on classifier performance," in *Proc. 25th Int. Conf. Knowl. Discovery Data Mining*, 1999, pp. 367–371.

[43] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, Dec. 2007.

[44] G. Krempl and V. Hofer, "Classification in presence of drift and latency," in *Proc. 11th Data Mining Workshops*, Dec. 2011, pp. 596–603.

[45] M. Krivko, "A hybrid model for plastic card fraud detection systems," *Expert Syst. Appl.*, vol. 37, no. 8, pp. 6070–6076, 2010.

[46] T.-Y. Liu, "Learning to rank for information retrieval," *Found. Trends Inf. Retr.*, vol. 3, no. 3, pp. 225–331, 2009.

[47] N. Mahmoudi and E. Duman, "Detecting credit card fraud by modified fisher discriminant analysis," *Expert Syst. Appl.*, vol. 42, no. 5, pp. 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," *Ann. Math. Statist.*, vol. 18, no. 1, pp. 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 Recognit.*, vol. 45, no. 1, pp. 521–530, 2012.

[50] K. Nishida and K. Yamauchi, "Detecting concept drift using statistical testing," in *Discovery Science*. Berlin, Germany: Springer, 2007, pp. 264–269.

[51] D. Olszewski, "Fraud detection using self-organizing map visualizing the user profiles," *Knowl.-Based Syst.*, vol. 70, pp. 324–334, Nov. 2014.

[52] C. Phua, V. Lee, K. Smith, and R. Gayler. (Sep. 2010). "A comprehensive survey of data mining-based fraud detection research." [Online]. Available: https://arxiv.org/abs/1009.6119

[53] J. Plasse and N. Adams, "Handling delayed labels in temporally evolving data streams," in *Proc. Int. Conf. Big Data*, 2016, pp. 2416–2424.

[54] J. T. Quah and M. Sriganesh, "Real-time credit card fraud detection using computational intelligence," *Expert Syst. Appl.*, vol. 35, no. 4, pp. 1721–1732, 2008.

[55] Y. Sahin, S. Bulkan, and E. Duman, "A cost-sensitive decision tree approach for fraud detection," *Expert Syst. Appl.*, vol. 40, no. 15, pp. 5916–5923, 2013.

[56] D. Sánchez, M. A. Vila, L. Cerda, and J. M. Serrano, "Association rules applied to credit card fraud detection," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3630–3640, 2009.

[57] J. C. Schlimmer and R. H. Granger, Jr., "Incremental learning from noisy data," *Mach. Learn.*, vol. 1, no. 3, pp. 317–354, 1986.

[58] B. Settles, "Active learning literature survey," *Univ. Wisconsin, Madison*, vol. 52, nos. 55–66, p. 11, 2010.

[59] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *J. Stat. Planning Inference*, vol. 90, no. 2, pp. 227–244, 2000.

[60] P. Sobhani and H. Beigy, *New Drift Detection Method for Data Streams*. Berlin, Germany: Springer, 2011.

[61] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proc. 7th Int. Conf. Knowl. Discovery Data Mining*, 2001, pp. 377–382.

[62] D. K. Tasoulis, N. M. Adams, and D. J. Hand, "Unsupervised clustering in streaming data," in *Proc. Int. Conf. Data Mining Workshops*, 2006, pp. 638–642.

[63] V. Van Vlasselaer *et al.*, "*APATE*: A novel approach for automated credit card transaction fraud detection using network-based extensions," *Decision Support Syst.*, vol. 75, pp. 38–48, Jul. 2015.

[64] S. Wang, L. L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1356–1368, May 2015.

[65] D. J. Weston, D. J. Hand, N. M. Adams, C. Whitrow, and P. Juszczak, "Plastic card fraud detection using peer group analysis," *Adv. Data Anal. Classification*, vol. 2, no. 1, pp. 45–62, 2008.

[66] C. Whitrow, D. J. Hand, P. Juszczak, D. J. Weston, and N. M. Adams, "Transaction aggregation as a strategy for credit card fraud detection," *Data Mining Knowl. Discovery*, vol. 18, no. 1, pp. 30–55, 2009.

[67] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, 1996.

[68] K. Yamazaki, M. Kawanabe, S. Watanabe, M. Sugiyama, and K.-R. Müller, "Asymptotic Bayesian generalization error when training and test distributions are different," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 1079–1086.

[69] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 114.

[70] B. Zadrozny, J. Langford, and N. Abe, "Cost-sensitive learning by cost-proportionate example weighting," in *Proc. Int. Conf. Data Mining*, Nov. 2003, pp. 435–442.

[71] V. Zaslavsky and A. Strizhak, "Credit card fraud detection using self-organizing maps," *Inf. Secur.*, vol. 18, p. 48, 2006.

[72] I. Žliobaitė. (Oct. 2010). "Learning under concept drift: An overview." [Online]. Available: https://arxiv.org/abs/1010.4784

Andrea Dal Pozzolo received the master's degree *(cum laude)* from the Faculty of Statistics, Universitá di Bologna, Bologna, Italy, in 2011, and the Ph.D. degree from the Machine Learning Group, Université Libre de Bruxelles, Brussels, Belgium, in 2015, where he was involved in machine learning and statistical techniques for fraud detection.

He is currently a consultant for major Banks and Insurance companies. His current research interests include unbalanced data streams, cost-sensitive learning, and concept drift.

Giacomo Boracchi received the M.S. degree in mathematics from the Università degli Studi di Milano, Milan, Italy, in 2004, and the Ph.D. degree in information technology from the Politecnico di Milano, Milan, in 2008, respectively.

He was a Researcher with the Tampere International Center for Signal Processing, Tampere, Finland, in 2004–2005. He is currently an Assistant Professor with the Dipartimento di Elettronica, Informazione e Bioingegneria of the Politecnico di Milano, Milan. His current research interests include learning methods for nonstationary environments as well as mathematical and statistical methods for image processing and analysis.

Dr. Boracchi was a recipient of the IBM Faculty Award in 2015 and the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS Outstanding Paper Award in 2016. He received the Nokia Visiting Professor Grant in 2017.

Olivier Caelen received the Ph.D. degree from the Machine Learning Group, Université Libre de Bruxelles, Brussels, Belgium, under the supervision of Prof. G. Bontempi.

After spending five years in the credit card fraud detection team, he is currently with the R&D High Processing & Volume Team, Worldline (an Atos company), Brussels, Belgium. His current research interests include machine learning techniques for anomaly and fraud detection.

Cesare Alippi (SM'94–F'06) is currently a Professor with the Politecnico di Milano, Milan, Italy, and the Università della Svizzera Italiana, Lugano, Switzerland. His current research interests include adaptation and learning in nonstationary environments and intelligence for embedded and cyberphysical systems.

Dr. Alippi is a member of the Administrative Committee of the IEEE Computational Intelligence Society, the Board of Governors of the International Neural Network Society, and the Board of Directors of the European Neural Network Society. He was a recipient of the Gabor Award from the International Neural Networks Society and the IEEE Computational Intelligence Society Outstanding TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS Paper Award in 2016 and the IEEE Instrumentation and Measurement Society Young Engineer Award in 2004.

Gianluca Bontempi (SM'12) received the master's degree (Hons.) in electronic engineering from the Politecnico of Milan, Milan, Italy, and the Ph.D. degree in applied sciences from the Université Libre de Bruxelles (ULB), Brussels, Belgium.

He is currently a Full Professor with the Computer Science Department, ULB, where he is the Co-Head with the ULB Machine Learning Group and the Director with the Interuniversity Institute of Bioinformatics in Brussels, Brussels. He was involved in research projects in academy and private companies all over Europe. He has authored or co-authored over 200 scientific publications and software for data mining and prediction, which was awarded in two international competitions. His current research interests include data mining, scalable machine learning, bioinformatics, and time series prediction.