Learned lessons in credit card fraud detection from a practitioner perspective

Article *in* Expert Systems with Applications · August 2014

DOI: 10.1016/j.eswa.2014.02.026

5 authors, including:

Andrea Dal Pozzolo
Université Libre de Bruxelles
8 PUBLICATIONS **520** CITATIONS
SEE PROFILE

Olivier Caelen
Orange Labs
58 PUBLICATIONS **867** CITATIONS
SEE PROFILE

Yann-Aël Le Borgne
Université Libre de Bruxelles
68 PUBLICATIONS **921** CITATIONS
SEE PROFILE

Gianluca Bontempi
Université Libre de Bruxelles
321 PUBLICATIONS **9,147** CITATIONS
SEE PROFILE

Learned lessons in credit card detection from a practitioner perspective

Submitted for Blind Review

Abstract

Credit-card fraud detection is an online problem where data are typically strongly unbalanced, arrive periodically in chunks and the distribution is not stationary. At the same time data are scarcely available for confidentiality issues, leaving unanswered many questions about which is the best strategy to deal with them.

In this paper we provide some answers from the practitioner's perspective by focusing on three crucial issues: unbalancedness, nonstationarity and assessment.

The analysis is made possible by two real credit card datasets provided by our industrial partner.

Keywords: Incremental Learning, Unbalanced data, Fraud detection

1. Introduction

Nowadays, enterprises and public institutions have to face a growing presence of fraud initiatives and need automatic systems to implement fraud detection. Automatic systems are essential since it is not always possible or easy for a human analyst to detect fraudulent patterns in transaction datasets, often characterized by a large number of samples, many dimensions and online updates. Also, since the number of fraudulent transactions is (fortunately) much smaller than the legitimate ones, the data distribution is unbalanced, i.e. skewed towards non-fraudulent observations. It is well known that many learning algorithms underperform when used for unbalanced dataset [1] and methods (e.g. resampling) have been proposed to improve their performances. Unbalancedness is not the only factor that determines the difficulty of a classification/detection task. Another influential factor is the amount of overlapping of the classes of interest due to limited

information that transaction records provide about the nature of the process [2].

Detection problems are typically addressed in two different ways. In the static learning setting, a detection model is periodically relearnt from scratch (e.g. once a year or month) since it is possible to store and process the entire dataset all at once. In the online learning setting, the detection model is updated as soon as new data is there. Though this strategy is the most adequate to deal with issues of non stationarity (e.g. due to the evolution of the spending behavior of the regular card holder or the fraudster), little attention has been devoted in the literature to the unbalanced setting.

Another problematic issue in credit card detection is the scarcity of available data due to confidentiality issues that give little chance to the community to share real datasets and assess existing techniques.

This paper aims to fill this gap by making an exhaustive comparison of a large number of algorithms and modeling techniques on two real datasets, focusing in particular on some open questions like: Which machine learning algorithm should be used? Is it enough to learn a model once a month or it is necessary to update the model everyday? How many transactions are sufficient to train the model? Should the data be analyzed in their original unbalanced form? If not, which is the best way to rebalance them? Which performance measure is the most adequate to asses results?

In this paper we address these questions with the aim of assessing their importance on real data and from a practitioner perspective. The paper starts by analyzing the fraud problem and provides a formalisation of the detection task. We then discuss existing alternative measures to classic classification metrics to be taken into consideration for the detection task. Then we compare three approaches for online learning in order to identify which informations are important to retain or to forget in a changing and non-stationary environment. Finally we show the impact of the rebalancing technique on the final performance. All the results are obtained by experimentation on two datasets of real credit card transactions provided by our industrial partner.

2. State of the art in credit card fraud detection

Credit card fraud detection is one of the most explored domains of fraud detection [3, 4, 5] and relies on the automatic analysis of recorded transactions to detect fraudulent behavior. Every time a credit card is used,

transaction data, composed of a number of attributes (e.g. credit card identifier, transaction date, recipient, amount of the transaction), are stored in the databases of the service provider. However a single transaction information is typically not sufficient to detect a fraud occurrence [3] and the analysis has to take place at the level of the credit card. At the card level it is possible to group transactions from the same card and learn behavioural models of individual cards. For each card, attributes like total spent per day, transaction number per week or average amount of a transaction, are extracted and analysed to detect fraudulent uses of the credit card [6].

2.1. Supervised Vs Unsupervised detection

In the fraud detection literature we encounter both supervised techniques that make use of the class of the transaction (e.g. genuine or fraudulent) and unsupervised techniques. Supervised methods assume that classifications of past transactions are available and reliable but are often limited to recognize fraud patterns that have already occurred [7]. On the other hand, unsupervised methods aim to detect those transactions that are most dissimilar from the norm by modeling the distribution of the majority of transactions.

The focus of this paper will be on supervised methods. In literature several supervised methods have been applied to fraud detection such as Neural networks [8], Rule-based methods (BAYES [9], RIPPER [10]) and tree-based algorithms (C4.5 [11] and CART [12]). It is well known however that an open issue is how to manage unbalanced class sizes since the legitimate transactions generally far outnumber the fraudulent ones.

2.2. Unbalanced problem

Learning from unbalanced datasets is a difficult task since most learning systems are not designed to cope with a large difference between the number of cases belonging to each class [13]. In the literature, traditional methods for classification with unbalanced dataset take advantage of sampling techniques to balance the dataset [1].

In particular we can distinguish between methods that operates at the data and algorithmic levels [14]. At the data level, balancing techniques are used as a pre-processing step to rebalance the dataset or to remove the noise between the two classes, before any algorithm is applied. At the algorithmic level, the classification algorithms themselves are adapted to deal with the minority class detection. In this article we focus on data level sampling and ensemble techniques.

Sampling techniques do not take into consideration any specific information in removing or adding observations from one class, yet they are easy to implement and to understand. *Undersampling* [15] consists in down-sizing the majority class by removing observations at random until the dataset is balanced. *SMOTE* [16] over-samples the minority class by generating synthetic minority examples in the neighborhood of observed ones. The idea is to form new minority examples by interpolating between examples of the same class. This has the effect of creating clusters around each minority observation.
Ensemble methods combine balancing techniques with a classifier to explore the majority and minority class distribution. *EasyEnsemble* is claimed in [17] to be better alternative to undersampling. This method learns different aspects of the original majority class in an unsupervised manner. This is done by creating different balanced training sets by *Undersampling*, learning a model for each dataset and then combining all predictions as in bagging.

2.3. Incremental learning

Static learning is the classical learning setting where the data are processed all at once in a single learning batch. *Incremental learning* instead interprets data as a continuous stream and processes each new instance "on arrival" [18]. In this context it is important to preserve the previously acquired knowledge as well as to update it properly in front of new observations (one-pass constraint). In incremental learning data arrives in chunks while static learning deals with a single dataset.

The problem of learning in the case of unbalanced data has been widely explored in the static learning setting[1, 15, 16, 17]. Learning from nonstationary data stream with skewed class distribution is however a relatively recent domain.

The problem of learning new information and retaining existing knowledge in nonstationary data streams is well known under the stability-plasticity dilemma [19]. Many of the techniques proposed [20, 21, 22] use ensemble classifiers in order to combine what is learnt from new observations and the knowledge acquired before. As fraud evolves over the time, the learning framework has to adapt to the new distribution. The classifier should be able to learn from a new fraud distributions and "forget" outdated knowledge. It becomes critical then to set the rate of forgetting in order to match the rate of change in the distribution [23]. The simplest strategy uses a constant forgetting rate, which boils down to consider a fix window of recent observations

to retrain the model. FLORA approach [24] uses a variable forgetting rate where the window is shrunk if a change is detected and expanded otherwise. The evolution of a class concept is called in literature *concept drift*.

Gao [25] proposes to store all previous minority class examples into the current training data set to make it less unbalanced and then to combine the models into an ensemble of classifiers. SERA [26] and REA [27] selectively accumulate old minority class observations to rebalance the training chunk. They propose two different methods (Mahalanobis distance and K nearest neighbours) in order to select the most relevant minority instances to include in the current chunk from the set of old minority instances.

The previous methods consist in *oversampling* the minority class of the current chunk by retaining old positive observations. Accumulation of previous minority class examples is of limited volume due to skewed class distribution, therefore oversampling does not increase a lot the chunk size.

3. Formalization of the learning problem

In this section, we formalize the credit card fraud detection task as a statistical learning problem. Let X_{ij} be a transaction number j of a card number i. We assume that the transactions are ordered in time such that if X_{iv} occurs before X_{iw} then $v < w$. For each transaction some basic information are available such as amount of the expenditure, the shop where it was performed, the currency, etc. However these variables however do not provide any information about the normal card usage. The normal behaviour of a card can be measured by using a set of historical transactions from the same card. Let X_{iN} be a new transaction and let $dt\left(X_{iN}\right)$ be the corresponding transaction date-time. Let T denote the time-frame of a set of historical transactions for the same card. X_{iN}^{H} is then the set of the historical transactions such that $X_{iN}^{H} = \{X_{ij}\}$, where $dt\left(X_{ij}\right) \neq dt\left(X_{iN}\right)$ and $dt\left(X_{ij}\right) \geq dt\left(X_{iN}\right) - T$. The card behaviour can be summarised using classical aggregation methods (e.g. *mean*, *max*, *min* or *count*) on the set X_{iN}^{H}. This means that it is possible to create new *aggregated* variables that can be added to the original transaction variables to include information of the card. Let $\{\mathcal{X}\}$ be the new set of transactions with aggregated variable. Each transaction $\mathcal{X}_{ij}$ is assigned a binary status $\mathcal{Y}_{ij}$ where $\mathcal{Y}_{ij} = 1$ when the transaction j is fraudulent otherwise $\mathcal{Y}_{ij} = 0$. Based on a set of historical transactions, the goal of a detection system is to learn $P\left(\mathcal{Y}|\mathcal{X}\right)$ and predict the class of a new transaction $\widehat{\mathcal{Y}}_{iN} \in (0, 1)$.

Credit card fraud detection has some specificities compared to classical machine learning problems. For instance, the continuous availability of new products in the market (like purchase of music on the Internet) changes the behaviour of the cardholders and consequently the distributions $P(\mathcal{X})$. At the same time the evolution of the types of frauds affects the class conditional probability distribution $P(\mathcal{Y}|\mathcal{X})$. As a result the joint distribution $P(\mathcal{X}, \mathcal{Y})$ is not stationary: this is is known as *concept-drift* []. Note that Gao [25] suggests that even when the concept drift is not detected, there is still a benefit in updating the models.

4. Performance measure

Fraud detection must deal with the following challenges: i) *timeliness* of decision (blocking a card immediately after the first fraud is not as blocking it after one week), ii) unbalanced class sizes (the number of frauds are relatively small compare to genuine transactions) and iii) cost structure of the problem (cost of not detecting one fraud is much higher than generating a false alert).

Let $\{\mathcal{Y}_0\}$ be the set of genuine transaction, $\{\mathcal{Y}_1\}$ the set of fraudulent transactions, $\{\widehat{\mathcal{Y}}_0\}$ the set of transactions predicted as genuine and $\{\widehat{\mathcal{Y}}_1\}$ the set of transactions predicted as fraudulent.

For a binary classification problem it is conventional to define a confusion matrix (Table 1).

	True Fraud ($\mathcal{Y}_1$)	True Genuine ($\mathcal{Y}_0$)
Predicted Fraud ($\widehat{\mathcal{Y}}_1$)	TP	FP
Predicted Genuine ($\widehat{\mathcal{Y}}_0$)	FN	TN

Table 1: Confusion Matrix

In an unbalanced class problem, it is well-known that quantities like TPR ($\frac{TP}{TP+FN}$), TNR ($\frac{TN}{FP+TN}$) and Accuracy ($\frac{TP+TN}{TP+FN+FP+TN}$) are misleading assessment measures [28]. Balanced Error Rate ($0.5 \times \frac{FP}{TN+FP} + 0.5 \times \frac{FN}{FN+TP}$) may be inappropriate too because of different costs of misclassification false negatives and false positives. Hand [29] considers also the use of Gini index and AUC as inappropriate, since these measures make an average of the misclassification cost of the two classes. In fraud detection we are interested in having high Precision ($\frac{TP}{TP+FP}$) and high Recall ($\frac{TP}{TP+FN}$). F-measure

$(2 \times \frac{Precision \times Recall}{Precision + Recall})$ gives equal importance to Precision and Recall into a metric ranging between 0 and 1 (the higher the better). This is often considered to be a relevant measure in fraud detection since it is able to combines Precision and Recall into a single metric. However this metric assumes that we are in a static environment where the observations are given all at once and we know all frauds.

4.1. Detection

The performance of a detection task (like fraud detection) is not necessarily well described in terms of classification [30]. In a detection problem what matters most is whether the algorithm can rank the few useful items (e.g. frauds) ahead of the rest. The focus then is not on predicting accurately each class but on returning a correct rank of the minority classes.

In detection teams like the one of our industrial partner, each time a fraud alert is generated by the detection system, this has to be checked by investigators before proceeding with actions (e.g. customer contact or card stop). Given the limited number of investigators it is possible to verify only a limited number of alerts. Therefore it is crucial to have the best ranking within the maximum number α of alerts that they can investigate. In this setting it is important to have the highest Precision within the first α alerts.

In the following we will denote as *PrecisionRank* the Precision within the α observations with the highest rank.

4.2. Cost analysis

In fraud detection the cost of frauds and investigations are known therefore it is possible to quantify the amount loss or potential loss. Starting from the confusion matrix (Table 1) we can assign a cost to each entry of the matrix. The cost of investigating an alert can be quantified by multiplying the average cost of an investigator per second and the average number of seconds spent for an alert. Let us define the cost of an alert as C_a. Since an alert is generate for a true positive (TP) or a false positive (FP), then the cost of a TP and FP can be set equal to C_a. The cost of a false negative can be obtained from the transaction amount ($TxAmt$) of the missed fraud. We assume that there is no cost in case of a true negative since no action is required apart from running the detection system. Taking in consideration all previous costs it is possible to define the total cost of detection ($Tcost$):

$$\text{Tcost} = \sum_{i=1}^{FN} TxAmt_i + (FP + TP) \times C_a$$

5. Strategies for incremental learning with unbalanced fraud data

The most conventional way to deal with sequential fraud data is to adopt a *static approach* (Figure 1) which creates once in a while a classification model and uses it as a predictor during a long horizon. Though this approach reduces the learning effort, its main problem resides in the lack of adaptivity which makes it insensitive to any change of distribution in the upcoming chunks.

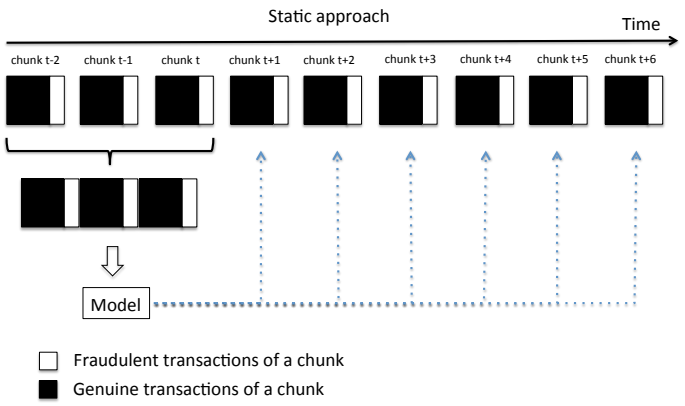


Figure 1: Static approach: a model is trained on $K = 3$ chunks and used to predict future chunks.

On the basis of the state-of-the-work described in Section, it is possible to conceive two alternative strategies to address both the incremental and the unbalanced nature of the fraud detection problem.

The first approach, denoted as the *updating* approach and illustrated in Figure 2), is inspired to Wang [31]. It uses a set of M models and a number K of chunks to train each model. Note that for $M > 1$ and $K > 1$ the training sets of the M models are overlapping.

This approach adapts to changing environment by forgetting chunks at a constant rate. The last M models are stored and used as a weighted voting

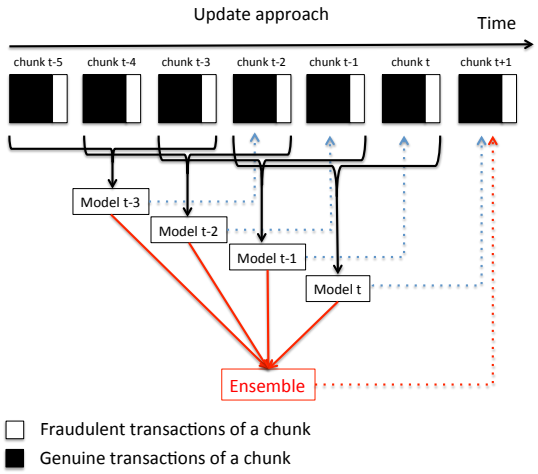


Figure 2: Updating approach for $K = 3$ and $M = 4$. For each new chunk a model is trained on the K latest chunks. Single models are used to predict the following chunk or can be combined into an ensemble.

ensemble, where the weight of the M^{th} model is returned by its PrecisionRank on the $M - 1$ previous chunks.

The second approach (denoted as the *forgetting genuine* approach and illustrated in Figure 3) is inspired to to Gao's work [25]. In order to mitigate the unbalanced effects, each time a new chunk is available, a model is learned on the genuine transactions of the previous K_{gen} chunks and all past fraudulent transactions. Since this approach leads to training sets which grow in size over the time a maximum training size is set to avoid overloading. Once this size is reached older observations are removed in favor of the more recent ones. An ensemble models is obtained by combining the last M models as in the update approach.

Note that in all these approaches (including the static one), a balancing technique (Section2.2) can be used to to reduce the skewness of the training set.

6. Experimental assessment

In this section we perform an extensive experimental assessment on the basis of real data (Section 6.1) in order to address common issues that the

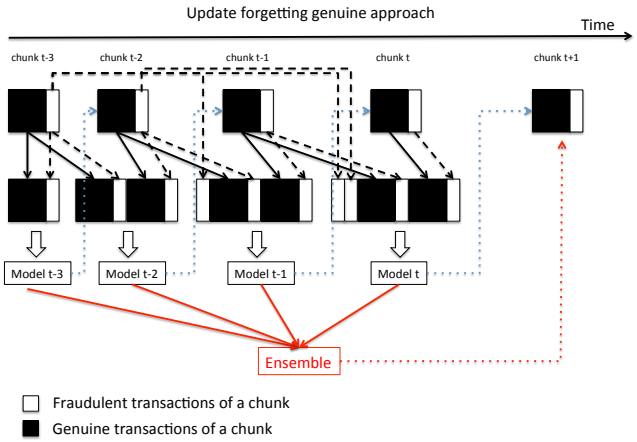


Figure 3: Forgetting genuine approach: for each new chunk a model is created by keeping all previous fraudulent transactions and a small set of genuine transactions from the last 2 chunks ($K_{gen} = 2$). Single models are used to predict the following chunk or can be combined into an ensemble ($M = 4$).

practitioner has to solve when facing large credit card fraud datasets (Section 6.2).

6.1. Datasets

Two credit card fraud datasets were provided by a payment service provider in Belgium. The first contains the logs of a subset of transactions from 2012 Feb, the 1st to 2012, Aug the 1st and the second concerns the period from 2012, Sep the 22th to 2013, May the 20th (details in Table 2). The two datasets share some basic variables such as transaction amount, point of sale, currency, country of the transaction and so on. We derived a set of aggregated variables from the basic ones (e.g. the average expenditure per week/day) in order to profile the user behavior.

Table 2: Fraudulent datasets

Dataset ID	Ndays	Nvar	Ntrx	Period
1	182	51	1341342	1Feb12 - 1Aug12
2	240	45	860886	22Sep12 - 20May13

These datasets are strongly unbalanced (the percentage of fraudulent

transactions is lower than 0.4%) and contain both categorical and continuous variables. In what follow we will consider that a chunk contains the set of daily transactions.

6.2. Learned lessons

The extensive experimental analysis allows to provide some answers to the most common questions that a practitioner could have when facing credit card fraud detection. The questions and the answers based on our experimental findings are detailed below.

6.2.1. Which algorithm and which training size is recommended in case of a static approach ?

The static approach (described in Section 5) is one of the most commonly used by practitioners because of its simplicity and rapidity. However, open questions remain about which learning algorithm should be used and the consequent sensitivity of the accuracy to the training size. We tested three different supervised algorithms: Random Forests (RF), Neural Network (NNET) and Support Vector Machine (SVM) with default parameters provided by the R software [32].

In order to assess the impact of the training set size (in terms of days/chunks) we carried out the predictions with different windows: 30, 60 and 90 days. We did not go over 90 days in order to preserve at least half of the chunks for testing. All training sets were rebalanced using undersampling at first. Figure 4 reports the average PrecisionRank and cost of detection for each month (EXPLAIN).

In both datasets, Random Forests clearly outperforms its competitors and, as expected, accuracy is improved by increasing the training size. Because of the significative superiority of Random Forests with respect to the other algorithms, in what follows we will limit to consider only this learning algorithm.

6.2.2. Is there an advantage in updating models?

Here we assess the advantage of adopting the *update* approach described in Section 5. Figure 5 reports the results for different values of K and M.

The strategies are called *daily* if a model is built every day, *weekly* if once a week or *15days* if every 15 days. When the strategy name includes *KModels* where K is a number, it means that an ensemble of K models is built for predictions, otherwise a single model is used.

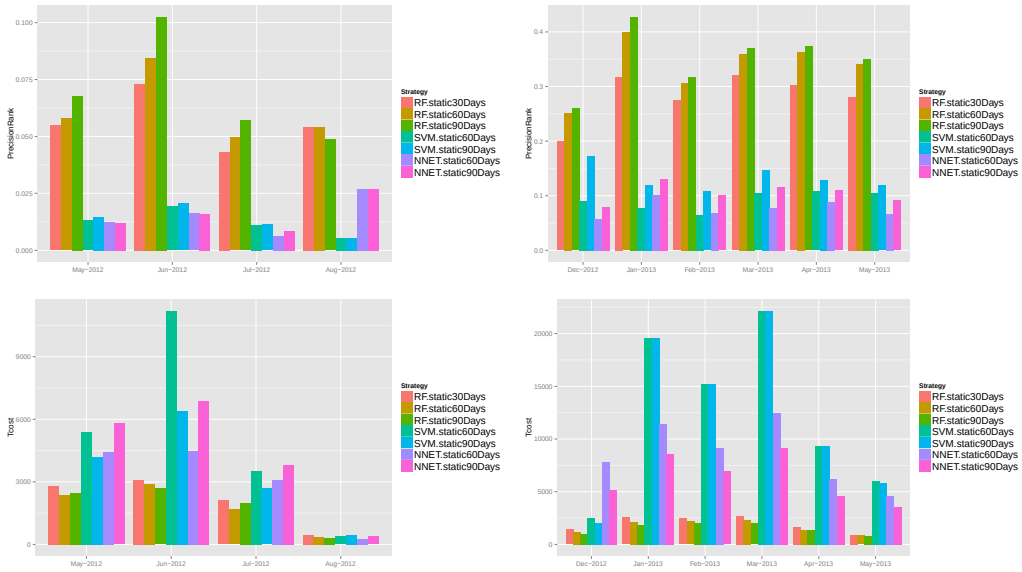


Figure 4: Static approach in terms of PrecisionRank (higher the better) and cost of detection (lower the better).

Though visually there is no clear winner, we can proceed by performing a significance Friedman test. Figure 6 shows the sum of the ranks from the Friedman test for each strategy in terms of PrecisionRank and cost of detection. The higher is the sum of ranks, the better is the accuracy. The white bars denote models which are significantly worse than the best ranked one (paired t-test based on the the ranks of the chucks). In both datasets, the best strategy with PrecisionRank is *daily.5Models.Update90Days*. It creates a new models at each chunk using previous $K = 90$ days and keeps the $M = 5$ models created for predictions. In the first dataset however this strategy is not statistically better than the two other ensemble approaches ranked as second and third. The same strategy is sub-optimal if the analysis is done in terms of cost of detection.

In both datasets, the strategies that use only the current chunk to build a model (*update1Day*) are coherently the worst. This confirms the result of previous analysis that a too short window of data (and consequently a very small fraction of frauds) is insufficient to learn a reliable model. For all future experiments we display the results obtained using the Friedman test.

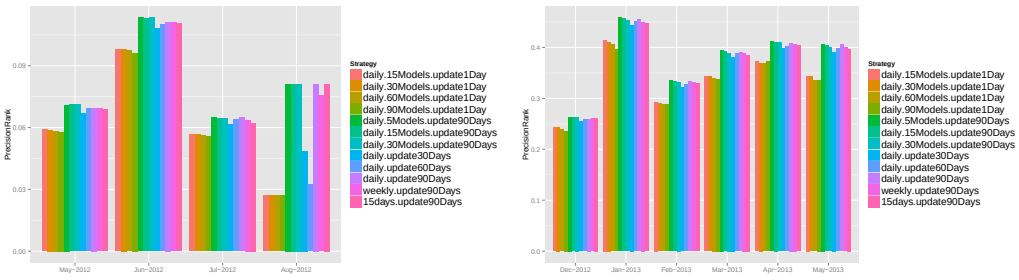
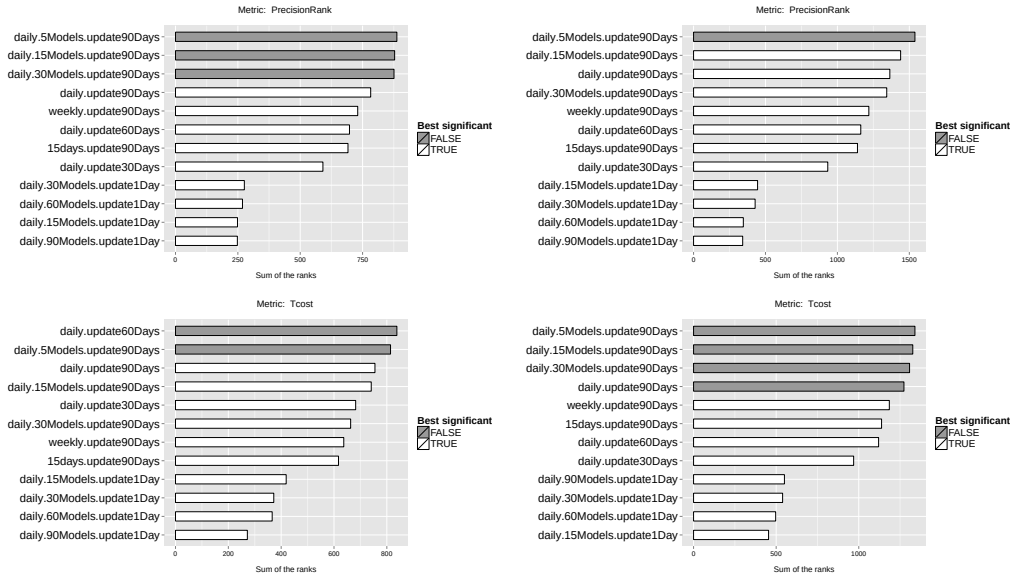


Figure 5: Performance with update approach.



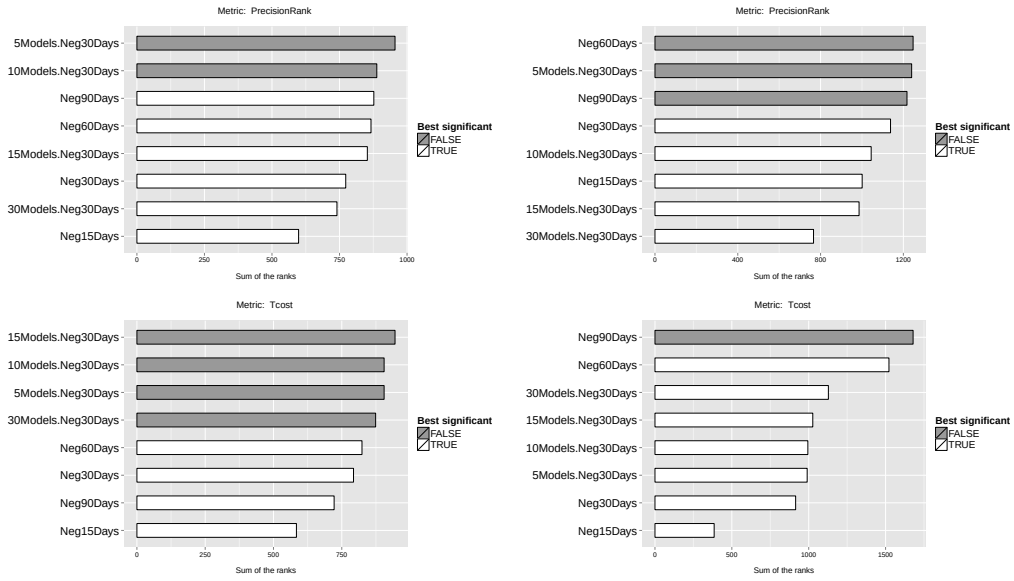
(a) Fraud dataset Feb-Aug

(b) Fraud dataset Sep-May

Figure 6: Sum of the ranks for different strategies with update approach.

6.2.3. Retaining old frauds is beneficial?

This section assesses the accuracy of the *forgetting* approach described in Section 5 whose rationale is to avoid the discard of old fraudulent observations. Figure 7 shows the sum of ranks for different training sets where we used a different number of days from which the genuine transactions are taken. The best strategy for the first dataset with PrecisionRank uses an ensemble of 5 models for each chunk. The same strategy is second for the other dataset, but is not significantly worse than the best. When we create an ensemble of M models, the window of models used is fix, which means at every new chunk the new model learnt replaces the M^{th} model in the models array where models are ordered by the time of creation. In terms of cost of detection, the best ensemble uses 15 models but is not significantly better than the one with 5. All these ensembles use only 30 days from which sampling the genuine transactions. For the second dataset however we see that the best strategy in terms of PrecisionRank and cost does not include an ensemble.

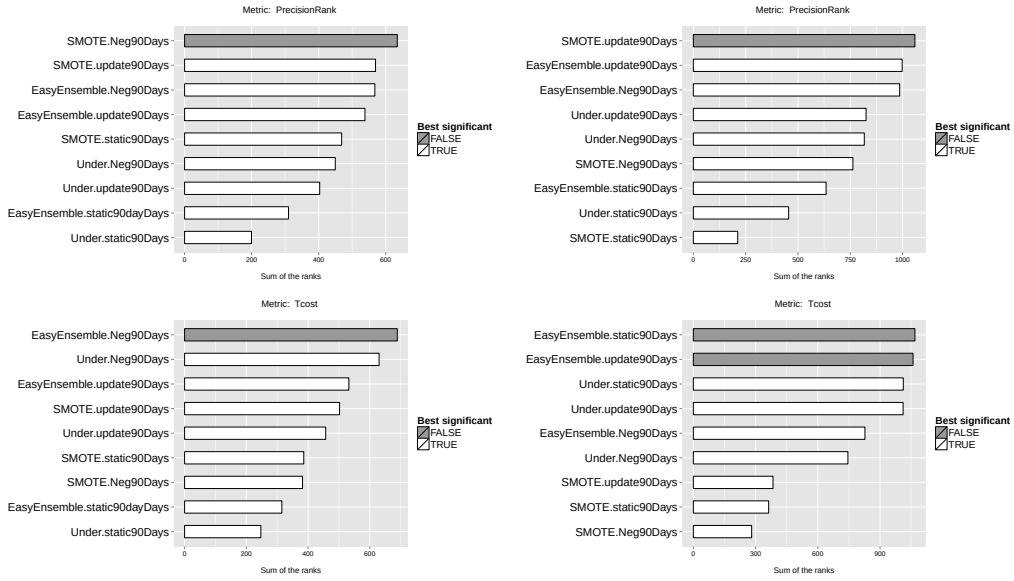


(a) Fraud dataset Feb-Aug (b) Fraud dataset Sep-May

Figure 7: Sum of the ranks for different strategies with different number of days for negative observations in the training set.

6.2.4. Do balancing techniques have an impact on accuracy?

So far we considered exclusively undersampling as balancing technique in our experiments. In this section we assess the impact of using alternative methods like SMOTE and EasyEnsemble. Experimental results (Figure 8) show that they both undersampling. In particular, the sets of best strategies in the two datasets include SMOTE for PrecisionRank and EasyEnsemble for Cost of detection.



(a) Fraud dataset Feb-Aug (b) Fraud dataset Sep-May

Figure 8: Sum of the ranks with different balancing methods.

6.2.5. Overall, which is the best strategy?

The large number of possible alternatives (i terms of learning classifier, balancing technique and incremental learning strategy) require a joint assessment of several combination in order to come up with a recommended approach. Figure 9 resume the entire set of experiments for the first dataset in terms of Precision Rank. The combinations of EasyEnsemble / SMOTE with the*forgetting* are able to overcome the ensemble strategies used in the update approach. EasyEnsemble is the best balancing technique also in terms of detection cost (Figure 10). This confirms than in online learning with unbalanced data, the adopted balancing strategy may play a major role. As

expected the static approach ranks low in both Figures 9 and 10 as it is not able to adapt to the changing distribution.

Strategy name	Algo	Unbalanced method	Approach	Update frequency	N Days	N models Ensemble	Sum of Ranks	Best Signif
SMOTE.Neg90Days	RF	SMOTE	ForgetGenuine	daily	90	1	1374.5	FALSE
EasyEnsemble.Neg90Days	RF	EasyEnsemble	ForgetGenuine	daily	90	1	1323.5	FALSE
SMOTE.update90Days	RF	SMOTE	Update	daily	90	1	1292.5	TRUE
EasyEnsemble.update90Days	RF	EasyEnsemble	Update	daily	90	1	1257.5	TRUE
Under.30Models.update90Days	RF	Under	Update	daily	90	30	1228.5	TRUE
Under.5Models.update90Days	RF	Under	Update	daily	90	5	1224	TRUE
Under.15Models.update90Days	RF	Under	Update	daily	90	15	1210	TRUE
Under.Neg90Days	RF	Under	ForgetGenuine	daily	90	1	1197	TRUE
Under.5Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	5	1112	TRUE
Under.update90Days	RF	Under	Update	daily	90	1	1107	TRUE
Under.30Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	30	1095.5	TRUE
Under.10Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	10	1073	TRUE
Under.15Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	15	1067.5	TRUE
SMOTE.static90Days	RF	SMOTE	Static	-	90	1	1063.5	TRUE
Under.weekly.update90Days	RF	Under	Update	weekly	90	1	1054.5	TRUE
Under.15days.update90Days	RF	Under	Update	15days	90	15	1004.5	TRUE
Under.static120Days	RF	Under	Static	-	120	1	952	TRUE
EasyEnsemble.static90Days	RF	EasyEnsemble	Static	-	90	1	817	TRUE
Under.static90Days	RF	Under	Static	-	90	1	686	TRUE
Under.static60Days	RF	Under	Static	-	60	1	536	TRUE
Under.static30Days	RF	Under	Static	-	30	1	451.5	TRUE
SVM.Under.static120Days	SVM	Under	Static	-	120	1	374.5	TRUE
NNET.Under.static120Days	NNET	Under	Static	-	120	1	305.5	TRUE
SVM.Under.static90Days	SVM	Under	Static	-	90	1	204.5	TRUE
SVM.Under.static60Days	SVM	Under	Static	-	60	1	184	TRUE
NNET.Under.static90Days	NNET	Under	Static	-	90	1	126	TRUE
NNET.Under.static60Days	NNET	Under	Static	-	60	1	114	TRUE

Figure 9: Comparison of all strategies using sum of ranks in all chunks for **PrecisionRank** in fraud dataset **Feb-Aug**.

Figures 11 and 12 compare all the strategies on the basis of accuracy for the second dataset. Here the best approach is the update approach which keeps 90 days for training each model. EasyEnsemble is once again the balancing techniques used within the winners. It is worth notice that ensembles which combines more than one model to predict are not superior to the predictions with a single model.

7. Conclusion

The paper has formalised the fraud detection problem and proposed PrecisonRank and total detection cost as the correct metrics for measuring the detection performance. To out knowledge, in literature few attempts [25, 33] have been done in combining research from incremental learning and unbalanced data. In our paper we propose a framework for fraud detection that can take advantage of any techniques already developed in the filed of unbalanced dataset. In particular we make use of two techniques (SMOTE and EasyEnsemble) and we show that they can both increase the performance of classical undersampling. We tested only three balancing techniques, but the framework can easily be extended to include any other unbalanced method.

Strategy name	Algo	Unbalanced method	Approach	Update frequency	N Days	N models Ensemble	Sum of Ranks	Best Signif
EasyEnsemble.Neg90Days	RF	EasyEnsemble	ForgetGenuine	daily	90	1	1487	FALSE
Under.Neg90Days	RF	Under	ForgetGenuine	daily	90	1	1416	FALSE
EasyEnsemble.update90Days	RF	EasyEnsemble	Update	daily	90	1	1225	TRUE
SMOTE.update90Days	RF	SMOTE	Update	daily	90	1	1167	TRUE
Under.5Models.update90Days	RF	Under	Update	daily	90	5	1167	TRUE
Under.30Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	30	1130	TRUE
Under.15Models.update90Days	RF	Under	Update	daily	90	15	1129	TRUE
Under.update90Days	RF	Under	Update	daily	90	1	1123	TRUE
Under.15Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	15	1088	TRUE
Under.30Models.update90Days	RF	Under	Update	daily	90	30	1070	TRUE
Under.15days.update90Days	RF	Under	Update	15days	90	15	1060	TRUE
Under.weekly.update90Days	RF	Under	Update	weekly	90	1	1046	TRUE
Under.10Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	10	979	TRUE
Under.5Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	5	933	TRUE
Under.static60Days	RF	Under	Static	-	60	1	925	TRUE
SMOTE.static90Days	RF	SMOTE	Static	-	90	1	883	TRUE
SMOTE.Neg90Days	RF	SMOTE	ForgetGenuine	daily	90	1	875	TRUE
Under.static120Days	RF	Under	Static	-	120	1	810	TRUE
EasyEnsemble.static90Days	RF	EasyEnsemble	Static	-	90	1	719	TRUE
Under.static90Days	RF	Under	Static	-	90	1	652	TRUE
Under.static30Days	RF	Under	Static	-	30	1	557	TRUE
SVM.Under.static120Days	SVM	Under	Static	-	120	1	542	TRUE
SVM.Under.static90Days	SVM	Under	Static	-	90	1	367	TRUE
NNET.Under.static60Days	NNET	Under	Static	-	60	1	347	TRUE
SVM.Under.static60Days	SVM	Under	Static	-	60	1	293	TRUE
NNET.Under.static120Days	NNET	Under	Static	-	120	1	262	TRUE
NNET.Under.static90Days	NNET	Under	Static	-	90	1	184	TRUE

Figure 10: Comparison of all strategies using sum of ranks in all chunks for **Cost of detection** in fraud dataset **Feb-Aug**.

Strategy name	Algo	Unbalanced method	Approach	Update frequency	N Days	N models Ensemble	Sum of Ranks	Best Signif
SMOTE.update90Days	RF	SMOTE	Update	daily	90	1	2670.5	FALSE
EasyEnsemble.Neg90Days	RF	EasyEnsemble	ForgetGenuine	daily	90	1	2646.5	FALSE
EasyEnsemble.update90Days	RF	EasyEnsemble	Update	daily	90	1	2619	FALSE
Under.5Models.update90Days	RF	Under	Update	daily	90	5	2522	TRUE
Under.5Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	5	2421	TRUE
Under.15Models.update90Days	RF	Under	Update	daily	90	15	2400	TRUE
Under.Neg90Days	RF	Under	ForgetGenuine	daily	90	1	2371.5	TRUE
Under.10Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	10	2316	TRUE
Under.update90Days	RF	Under	Update	daily	90	1	2298.5	TRUE
Under.30Models.update90Days	RF	Under	Update	daily	90	30	2246.5	TRUE
Under.15Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	15	2213.5	TRUE
SMOTE.Neg90Days	RF	SMOTE	ForgetGenuine	daily	90	1	2144.5	TRUE
Under.weekly.update90Days	RF	Under	Update	weekly	90	1	2098	TRUE
Under.30Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	30	1966.5	TRUE
Under.15days.update90Days	RF	Under	Update	15days	90	15	1956	TRUE
Under.static120Days	RF	Under	Static	-	120	1	1725.5	TRUE
EasyEnsemble.static90Days	RF	EasyEnsemble	Static	-	90	1	1701	TRUE
Under.static90Days	RF	Under	Static	-	90	1	1454	TRUE
Under.static60Days	RF	Under	Static	-	60	1	1251	TRUE
Under.static30Days	RF	Under	Static	-	30	1	970	TRUE
SMOTE.static90Days	RF	SMOTE	Static	-	90	1	833.5	TRUE
SVM.Under.static120Days	SVM	Under	Static	-	120	1	722	TRUE
SVM.Under.static90Days	SVM	Under	Static	-	90	1	493	TRUE
NNET.Under.static120Days	NNET	Under	Static	-	120	1	436	TRUE
NNET.Under.static90Days	NNET	Under	Static	-	90	1	409.5	TRUE
SVM.Under.static60Days	SVM	Under	Static	-	60	1	280	TRUE
NNET.Under.static60Days	NNET	Under	Static	-	60	1	194.5	TRUE

Figure 11: Comparison of all strategies using sum of ranks in all chunks for **PrecisionRank** in fraud dataset **Sep-May**.

Strategy name	Algo	Unbalanced method	Approach	Update frequency	N Days	N models Ensemble	Sum of Ranks	Best Signif
EasyEnsemble.update90Days	RF	EasyEnsemble	Update	daily	90	1	2624.5	FALSE
EasyEnsemble.static90Days	RF	EasyEnsemble	Static	-	90	1	2606	FALSE
Under.5Models.update90Days	RF	Under	Update	daily	90	5	2601.5	FALSE
Under.15Models.update90Days	RF	Under	Update	daily	90	15	2579	FALSE
Under.30Models.update90Days	RF	Under	Update	daily	90	30	2561.5	FALSE
Under.update90Days	RF	Under	Update	daily	90	1	2553.5	FALSE
Under.static90Days	RF	Under	Static	-	90	1	2522	TRUE
Under.weekly.update90Days	RF	Under	Update	weekly	90	1	2428	TRUE
Under.15days.update90Days	RF	Under	Update	15days	90	1	2393	TRUE
Under.static120Days	RF	Under	Static	-	120	1	2290	TRUE
Under.static60Days	RF	Under	Static	-	60	1	2277	TRUE
EasyEnsemble.Neg90Days	RF	EasyEnsemble	ForgetGenuine	daily	90	1	2149	TRUE
Under.static30Days	RF	Under	Static	-	30	1	2095	TRUE
Under.Neg90Days	RF	Under	ForgetGenuine	daily	90	1	2030	TRUE
SVM.Under.static120Days	SVM	Under	Static	-	120	1	1850	TRUE
SMOTE.update90Days	RF	SMOTE	Update	daily	90	1	1370	TRUE
SMOTE.static90Days	RF	SMOTE	Static	-	90	1	1212	TRUE
SMOTE.Neg90Days	RF	SMOTE	ForgetGenuine	daily	90	1	1089	TRUE
Under.30Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	30	1065.5	TRUE
Under.5Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	5	1049	TRUE
Under.10Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	10	1000.5	TRUE
Under.15Models.Neg30Days	RF	Under	ForgetGenuine	daily	30	15	996	TRUE
NNET.Under.static90Days	NNET	Under	Static	-	90	1	712	TRUE
NNET.Under.static60Days	NNET	Under	Static	-	60	1	485	TRUE
NNET.Under.static120Days	NNET	Under	Static	-	120	1	383	TRUE
SVM.Under.static90Days	SVM	Under	Static	-	90	1	223.5	TRUE
SVM.Under.static60Days	SVM	Under	Static	-	60	1	214.5	TRUE

Figure 12: Comparison of all strategies using sum of ranks in all chunks for **Cost of detection** in fraud dataset **Sep-May**.

Our framework violates the classical setting of incremental learning by retaining information from previous chunks. However the strategies proposed make use of small sets of transactions from previous chucks. The experimental part has shown that in online learning, when the data is skewed towards one class it is importance of maintaining previous examples from the minority class in order to learn a better model. This has the effect of oversampling the minority class in the current chunk but it is of limited impact given the small number of frauds. The second and third approaches proposed differ essentially in the amount of minority class retained in time. The latter accumulated more frauds leading to less skewed training set.

In [25, 33], an ensemble is created by combing models learnt on single chunks. Our experiments have showed that by combing information from the current and previous chunks we are able to increase the performance of each model included in the ensemble. The ensemble of models have proved to increase the performance with undersampling, but are not better that a single models where SMOTE or EasyEnsemble is used before training.

Our framework addresses the problem of non-stationary in data streams by creating a new model at each new chunk and using a fix windows of chunks in the training set. This approach has showed better results than updating the models at a lower frequency. Future work will focus on improving the combination of unbalanced techniques in the case of online learning.

References

[1] N. Japkowicz, S. Stephen, The class imbalance problem: A systematic study, Intelligent data analysis 6 (2002) 429–449.

[2] R. C. Holte, L. E. Acker, B. W. Porter, et al., Concept learning and the problem of small disjuncts, in: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, volume 1, Citeseer, 1989.

[3] R. Bolton, D. Hand, et al., Unsupervised profiling methods for fraud detection, Credit Scoring and Credit Control VII (2001) 235–255.

[4] R. Brause, T. Langsdorf, M. Hepp, Neural data mining for credit card fraud detection, in: Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on, IEEE, 1999, pp. 103–106.

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

[6] C. Whitrow, D. J. Hand, P. Juszczak, D. Weston, N. M. Adams, Transaction aggregation as a strategy for credit card fraud detection, Data Mining and Knowledge Discovery 18 (2009) 30–55.

[7] R. Bolton, D. Hand, Statistical fraud detection: A review, Statistical Science (2002) 235–249.

[8] J. Dorronsoro, F. Ginel, C. Sgnchez, C. Cruz, Neural fraud detection in credit card operations, Neural Networks, IEEE Transactions on 8 (1997) 827–834.

[9] P. Clark, T. Niblett, The cn2 induction algorithm, Machine learning 3 (1989) 261–283.

[10] W. W. Cohen, Fast effective rule induction, in: MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-, MORGAN KAUFMANN PUBLISHERS, INC., 1995, pp. 115–123.

[11] J. Quinlan, C4. 5: programs for machine learning, volume 1, Morgan kaufmann, 1993.

[12] L. Olshen, C. Stone, Classification and regression trees, Wadsworth International Group (1984).

[13] G. Batista, A. Carvalho, M. Monard, Applying one-sided selection to unbalanced datasets, MICAI 2000: Advances in Artificial Intelligence (2000) 315–325.

[14] N. V. Chawla, N. Japkowicz, A. Kotcz, Editorial: special issue on learning from imbalanced data sets, ACM SIGKDD Explorations Newsletter 6 (2004) 1–6.

[15] C. Drummond, R. Holte, et al., C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling, in: Workshop on Learning from Imbalanced Datasets II, Citeseer, 2003.

[16] N. Chawla, K. Bowyer, L. Hall, W. Kegelmeyer, Smote: synthetic minority over-sampling technique, Arxiv preprint arXiv:1106.1813 (2011).

[17] X. Liu, J. Wu, Z. Zhou, Exploratory undersampling for class-imbalance learning, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 39 (2009) 539–550.

[18] N. C. Oza, Online bagging and boosting, in: Systems, man and cybernetics, 2005 IEEE international conference on, volume 3, IEEE, 2005, pp. 2340–2345.

[19] S. Grossberg, Nonlinear neural networks: Principles, mechanisms, and architectures, Neural networks 1 (1988) 17–61.

[20] R. Polikar, L. Upda, S. Upda, V. Honavar, Learn++: An incremental learning algorithm for supervised neural networks, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 31 (2001) 497–508.

[21] W. N. Street, 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, ACM, 2001, pp. 377–382.

[22] S. Chen, H. He, K. Li, S. Desai, Musera: multiple selectively recursive approach towards imbalanced stream data mining, in: Neural Networks

(IJCNN), The 2010 International Joint Conference on, IEEE, 2010, pp. 1–8.

[23] L. I. Kuncheva, Classifier ensembles for changing environments, in: Multiple classifier systems, Springer, 2004, pp. 1–15.

[24] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Machine learning 23 (1996) 69–101.

[25] J. Gao, W. Fan, J. Han, On appropriate assumptions to mine data streams: Analysis and practice, in: Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on, IEEE, 2007, pp. 143–152.

[26] S. Chen, H. He, Sera: selectively recursive approach towards nonstationary imbalanced stream data mining, in: Neural Networks, 2009. IJCNN 2009. International Joint Conference on, IEEE, 2009, pp. 522–529.

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

[28] F. Provost, Machine learning from imbalanced data sets 101, in: Proceedings of the AAAI2000 Workshop on Imbalanced Data Sets, 2000.

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

[30] G. Fan, M. Zhu, Detection of rare items with target, Statistics and Its Interface 4 (2011) 11–17.

[31] H. Wang, W. Fan, P. S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2003, pp. 226–235.

[32] R Development Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2011. URL: `http://www.R-project.org/`, ISBN 3-900051-07-0.

[33] G. Ditzler, R. Polikar, N. Chawla, An incremental learning algorithm for non-stationary environments and class imbalance, in: Pattern Recognition (ICPR), 2010 20th International Conference on, IEEE, 2010, pp. 2997–3000.