Data Min Knowl Disc (2009) 18:30–55
DOI 10.1007/s10618-008-0116-z

Transaction aggregation as a strategy for credit card fraud detection

C. Whitrow · D. J. Hand · P. Juszczak · D. Weston · N. M. Adams

Received: 22 August 2007 / Accepted: 15 July 2008 / Published online: 30 July 2008

Abstract The problem of preprocessing transaction data for supervised fraud classification is considered. It is impractical to present an entire series of transactions to a fraud detection system, partly because of the very high dimensionality of such data but also because of the heterogeneity of the transactions. Hence, a framework for transaction aggregation is considered and its effectiveness is evaluated against transaction-level detection, using a variety of classification methods and a realistic cost-based performance measure. These methods are applied in two case studies using real data. Transaction aggregation is found to be advantageous in many but not all circumstances. Also, the length of the aggregation period has a large impact upon performance. Aggregation seems particularly effective when a random forest is used for classification. Moreover, random forests were found to perform better than other classification methods, including SVMs, logistic regression and KNN. Aggregation also has the advantage of not requiring precisely labeled data and may be more robust to the effects of population drift.

Keywords Fraud detection · Supervised classification · Credit cards · Preprocessing

Responsible editor: M. J. Zaki.

C. Whitrow (✉) · D. J. Hand · P. Juszczak · D. Weston
Institute for Mathematical Sciences, Imperial College,
London, UK
e-mail: c.whitrow@imperial.ac.uk

D. J. Hand · N. M. Adams
Department of Mathematics, Imperial College,
London, UK

1 Introduction

In this work, we consider the problem of identifying whether a credit or debit card account has been compromised by fraud. That is, we are interested in whether the account has been subject to fraudulent activity. There are two main levels on which we may approach this question: transaction level and account level. Banks usually implement some form of transaction classification system, often a set of rules, which raise alerts on individual transactions which are considered suspicious. A transaction might arouse suspicion if, for example, it is for a large amount of money and with a particular type of merchant (e.g. online bookmaker) at a certain time of day. This type of fraud detection strategy considers only transactions in isolation from each other. Neither the previous history of the associated account, nor other transactions are taken into consideration. It may be, for example, that a particular combination of transactions (maybe online gambling in combination with theater ticket bookings) is a better indicator of fraud than either transaction in isolation. Also, the occurrence of two transactions in geographically distant locations at nearly the same time (called a 'collision' or 'high velocity' event) is an often-used indicator of fraud. Moreover, it may also be that the frequency or volume of transactions is more important for fraud detection than the characteristics of any individual transaction.

In addition, some banks implement 'behavioral models' which aim to characterize the legitimate transaction behavior of each individual account over a period of time. For example, customer X only uses her credit card for retail purchases at supermarkets and high street shops, whilst customer Y tends to use his card for online purchases, especially at bookmakers. If, at some future time, online transactions start to appear on customer X's account, this change in behavior may be considered suspicious, as it is not characteristic of that customer's previous behavior. Behavioral models only consider the previous history of each account but do not attempt to identify global patterns of fraudulent behavior; they only try to detect changes in behavior. These models are unsupervised classifiers, examples of which have been discussed by Fawcett and Provost (1997) and Bolton and Hand (2001). The problem with this approach is that a change in behavior may not be due to fraud.

Given that neither behavioral models nor transaction-level classification are foolproof strategies for detecting fraud, it is important to find other strategies which may be able to ameliorate their weaknesses. All these strategies will ultimately be used in parallel, together with any existing rule-based detection system (e.g. for collision detection). An obvious extension to transaction-level classification is to aggregate information over a succession of transactions, or a period of time. This is indeed the approach often taken by designers of commercial fraud detection systems, which is one reason why we are examining it in this paper. For example, we may simply count the number of transactions at a particular type of merchant, or take the sum of the value of all online transactions over the past week or day. Whilst we may expect such transaction aggregation to result in better fraud detection than transaction-level classification, it is not clear how much advantage can be obtained by this strategy. Also, in the transition from transaction classification to account level aggregates, a lot of information is being discarded, especially concerning the order of transactions. In addition, we no longer have a transaction-level fraud/non-fraud signal. Instead, we

have a label which tells us only that there is some fraud among a set of transactions, some of which are also likely to be genuine. To put it another way, we have a signal identifying the account 'status' as fraudulent or not within an extended time window covered by the aggregation period. The contribution of a single fraud transaction is diluted when aggregation is used. However, if the aggregated data records are updated continuously (with every transaction) then it is still possible to identify fraud as soon as it occurs.

Thus, transaction aggregation is also not without problems, which become greater as the aggregation period lengthens. Our aim in this study is first to assess whether transaction aggregation may result in better fraud detection than transaction-level classification. We would also like to quantify this advantage, if any, and see how it varies with the length of the aggregation period. In doing so, we are following a two-class classification approach, using real historical labeled data from two collaborating banks. Understandably, there is relatively little previously published material on credit card fraud detection methods, although there are commercial products, which use a variety of the strategies outlined above (e.g. Fair 2007). As far as the authors are aware, there is no published work addressing the question of how useful transaction aggregation is compared to using individual transactions in a two-class classification framework.

For a recent survey of the general area of fraud detection, see Kou et al. (2004). Also see Bolton and Hand (2002) and Provost (2002) for a discussion of the problems particular to fraud detection which statistical techniques should seek to address. Chan et al. (1999) develop a boosting algorithm which uses a cost-weighted performance measure specifically designed to optimize classifiers for fraud detection. Although we do not make use of this algorithm here, we do use a cost-weighted measure for performance assessment (Hand et al. 2008). This is necessary, in the light of criticism of the conventional measures, such as those based on the ROC curve, when these are applied in a fraud detection context (see e.g. Hand 2005).

Previous published approaches to credit card transaction fraud, based on supervised classification, concentrate on classifying individual transactions rather than account level detection. Ghosh and Reilly (1994) use covariates from a medium-term (8–10 weeks of aggregated data) account level profile within a transaction-level RBF-type neural network classifier. Dorronsoro et al. (1997) use some short-term transaction history, although it is not very clear what this consists of and most of the inputs to their classifiers are from the transaction records. For other examples of neural network based transaction-level classifiers, see Aleskerov et al. (1997) and Brause et al. (1999). More recently, Maes et al. (2002) have compared neural classifiers with Bayesian networks for transaction classification. For an approach to credit card application fraud using case-based reasoning, see Wheeler and Aitken (2000).

In this paper we address the general problem of how best to present the data to a fraud classifier by developing a framework for aggregating transaction records. We investigate how this affects detection performance. Our emphasis is upon account-level detection rather than the classification of individual transactions. Also, in order to keep a general perspective, we use a broad variety of classifiers. The following section explains our strategy in more detail, including classification methods and

performance measurement. After that, we discuss the data itself, before going on to present our results and conclusions.

2 Methodology

2.1 Formal description of data attributes

The data for each transaction is a vector of characteristics, **x**. This encodes various aspects of the transaction: typically the amount (i.e. value), type (e.g. payment, refund, balance enquiry), merchant type (where applicable, e.g. supermarket), channel (e.g. ATM or POS terminal) and verification mode (e.g. chip and PIN, magnetic stripe etc.), among others. Fraud may be identified at the transaction level, or in some cases one may know that there is fraud on an account but not know for sure which transactions were fraudulent.

The status of an account may be regarded as a latent time-dependent binary variable, S_t. This indicates whether the account has suffered a fraudulent transaction at or prior to time t. The subscript t may be regarded either as an actual time or as the index of a transaction, since these are ordered in time. The task of a fraud detection system is to identify the latent state S_t for any t. By convention, we choose $S_t = 1$ to represent a fraud state (i.e. the account has been compromised), whilst $S_t = 0$ represents a normal non-fraud account state. For most accounts $S_t = 0\ \forall t$. However, for some accounts, the state changes from $S_t = 0$ (for $t < t_f$) to $S_t = 1$ (for $t \geq t_f$). This is an irreversible step change which needs to be identified, as soon after t_f as possible.

2.2 Why aggregate?

One of the chief dilemmas facing any fraud detection algorithm or system is determining at what point an alarm should be raised. Information about the status of each account is continually being updated as new transactions occur. This new information should allow better discrimination between fraudulent and non-fraudulent behavior. Three suspicious transactions in a row is surely more likely to be indicative of fraud than just one transaction in isolation. Thus, as information from transactions accumulates, the system should become more confident in its diagnosis of the underlying (latent) state, S_t, of the account.

However, the marginal value of new information may diminish as time passes. Aggregating 101 transactions is not likely to be much more useful than aggregating 100. In fact, the consequent loss of temporal resolution may obscure any fraudulent signal in the data. Worse still, as time passes, an account in a fraud state (i.e. one which has been compromised and is subject to fraudulent activity) is potentially incurring costs in the form of fraudulent transactions. Clearly, it is better to raise an alarm as early as possible in order to limit these losses. On the other hand, if one raises an early alarm, using relatively little information, confidence is likely to be low and there will be a high false alarm rate, which also leads to higher costs and may also result in many frauds going undetected. The question may therefore be phrased as: how much

information is needed before a decision about the status of an account can be made with confidence?

It is hoped that using transaction information accumulated over time will result in better fraud discrimination than one can obtain at the level of isolated transactions. Certain combinations of transactions may together be suspicious, even though those same transactions in isolation are not. For example, one may find that a mobile phone top-up followed by a series of online electronics purchases is peculiarly characteristic of fraudulent behavior, whilst those individual transactions alone are not. Such a combination of transactions might be called a 'fraud signature'. That is, there may be useful information in interactions between transactions and such information must be gathered over some interval of time. On the other hand, if this interval is too long then the information will be too diluted and it may even be too late to do anything. Hence, we expect there will be some optimal length for the aggregation period.

2.3 How to aggregate

The foregoing considerations lead us to perform a transformation from the transaction level to the account level so that each observation consists of an account level summary of transaction data, $\mathbf{y} = \Phi(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, where $\mathbf{x}_1, \ldots, \mathbf{x}_n$ represent a time-ordered series of recent transactions. We refer to $\mathbf{y}$ as an 'activity record' (in contrast to a transaction record). The number, n, of transactions could be fixed or we could include all transactions in the past up to a fixed time limit, so n may vary between accounts. The function Φ is the pre-processing transform, which may take many forms. We have chosen Φ such that $\mathbf{y}$ is a set of summary statistics on the transactions, $\mathbf{x}_1, \ldots, \mathbf{x}_n$ so that their order does not matter. That is, Φ is invariant to permutations of its arguments in this case (although it need not be so, e.g. if one considers a function such as temporal correlation in order to identify a trend). Of course, this means that we will be unable to identify a pattern of fraudulent transactions in which such ordering is important. This is unfortunate but we must somehow reduce the dimensionality of the data, which inevitably means losing some information. In proceeding this way, we hope that ordering information is not vital for fraud detection or else we will rely on some other technique to discover patterns where ordering is important. We will see that the classifiers built using aggregated data are capable of good performance and this helps to justify our approach, although it is likely that supplementary methods will be needed to improve performance further.

Several salient aspects of the transaction record, $\mathbf{x}$, were identified as being relevant to fraud detection by exploratory data analysis. For example, one statistic was the number of POS transactions (i.e. at point-of-sale terminals). Another statistic was the total value of these transactions. Similarly, other statistics were defined as counts or total value of various other types of transaction. To give another example, transactions verified by magnetic stripe are more likely to be fraudulent than chip and PIN transactions. Hence, two other statistics were defined as the total count and total value of transactions verified by magnetic stripe.

Transactions were also segmented by merchant code into groups of merchants with differing fraud risk rates. The merchant field has about 500 or so possible values.

Table 1 Example transaction records

Amt	Time	Service_id	Entry_mode	Merchant
50.00	11:25	ATM		
12.99	13:00	POS	CNP	1550
33.71	13:30	POS	PIN	4500
5.20	13:45	POS	Mag	1105
100.00	18:15	ATM		

Table 2 Example aggregated record

	Pos	ATM	CNP	PIN	Mag	Mgrp1	Mgrp3	TODx	TODy
Amt	51.90	150.00	12.99	33.71	5.20	18.19	33.71	−91.23	−110.74
Num	3	2	1	1	1	2	1	−3.71	−1.93

These were first grouped into 24 similar 'super-categories' (this leads to the creation of 48 variables) based upon a separate analysis using Laplace-smoothed estimates of the fraud rates within the categories. We attempted to create groups with fairly homogeneous fraud rate and a reasonably large number of total observations whilst maintaining the ordering of the codes. In this way, it was possible to derive count and total value statistics for merchant groups, as well as other groups of transactions. An explicit example may help to clarify the process. Consider a set of transactions as specified in Table 1.

These simplified transactions may be aggregated to create a single record shown in Table 2.

As can be seen, there are 2 ATM transactions, along with 3 at point-of-sale terminals. This fact is summarized in the second line of the aggregated record, where the number of different types of transactions is given. 'Entry_mode' and 'Merchant' information is relevant only for the POS transactions. Here, 'CNP' means 'card not present' (as is usual for telephone or internet transactions). The 3 different entry modes are aggregated separately and there is only 1 transaction of each type. The first line of the aggregated record gives the total amount of each type of transaction. For example, the two ATM withdrawals amounted to £150 in total, where as the POS transactions add up to £51.90 all together. Although there are 3 merchant codes, these have been put into 2 different groups (according to a separate analysis), with the codes 1105 and 1550 assigned to merchant group 1 and code 4500 assigned to group 2 (along with other codes which we do not mention here). The total amount of the two transactions in group 1 is £18.19. Finally, there are two 'time-of-day' aggregation variables which are explained in detail below.

Formally, $\mathbf{x}$ is first mapped to a vector of binary flags, $\mathbf{z} = Z(\mathbf{x})$, so that $\mathbf{z}$ indicates whether $\mathbf{x}$ is a member of a particular class of transactions (e.g. POS transactions, transactions at a merchant belonging to some particular group, transactions verified by magnetic stripe, etc.). Then, for each class of transactions, the summary statistics are the count and total value. Thus, for a particular account, $\mathbf{y} = \left(\sum_{i=1}^{n} \mathbf{z}_i, \sum_{i=1}^{n} a_i \mathbf{z}_i\right)$, where a_i is the amount (monetary value) of the ith transaction. There were also a couple of features that did not quite fit this general scheme, although they were defined very

similarly. For example, the time of day of each transaction was considered important, as it was noticed that late night and early morning transactions tended to be associated more with fraud. The time of each transaction was coded as a pair of variables, $\zeta_1 = \cos(2\pi t/T)$ and $\zeta_2 = \sin(2\pi t/T)$ where t is the time of day and T is the length of the day. Hence midday becomes the vector $(-1, 0)$ and midnight is $(1, 0)$. The vector $(0, 1)$ represents 6 am and $(0, -1)$ is 6 pm. This coding avoids the arbitrary discontinuity which otherwise occurs at midnight and is more efficient than using 'bins' for different times of day. The summary statistics are then still defined as above, treating ζ_1 and ζ_2 just like elements of the indicator vector $\mathbf{z}$, which is now allowed to take real values. The aggregated time-of-day variables are then defined as $\mathbf{t} = \left(\sum_{i=1}^{n} \boldsymbol{\zeta}_i, \sum_{i=1}^{n} a_i \boldsymbol{\zeta}_i\right)$, in much the same way as the other aggregated variables. The kind of pre-processing described here is commonly used in commercial fraud detection systems and is known to work well in many commercial applications, from the authors' own experience.

In general, we do not have to restrict attention to the counts and amounts, a_i, but could calculate sample moments or other statistics for any attribute we might consider interesting, such as the variance of geographical locations or maximum difference between successive amounts. The possibilities for features which may be included in $\mathbf{y}$ are endless. We have merely chosen a simple scheme for this study, based largely on measures of transaction volume. Other aggregation schemes are equally valid and can be incorporated within the same framework.

Note also that we do not require accurate labels (fraud/non-fraud) for all transactions. Indeed, it is often the case that banks cannot identify with certainty which transactions are fraudulent and can say with confidence only that there was some fraud on a certain date or dates. In such cases, transaction aggregation is particularly useful, since it is only necessary to specify a time (t_f) at which the status of an account changes from non-fraud to fraud. An 'activity record', $\mathbf{y} = \Phi(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, is allocated to the fraud class if *any* of the transactions $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ in its argument has a time later than t_f. An 'activity record' may therefore be defined to have a 'generation time' equal to the time of the last transaction in its argument. This formulation and definition of fraud status is also robust to misclassifications of individual transactions. Furthermore, in practice it is expected that there may be legitimate transactions mixed with fraud transactions on an account (in a fraud state). Any successful detection system will have to identify fraud states in spite of the possible presence of legitimate transactions. Indeed, legitimate transactions may sometimes be useful in identifying fraud states. For example, the near simultaneous use of a card in very different locations probably implies fraud, even though one does not know which of the transactions is actually fraudulent. Considerations such as these provide strong justification for the aggregation strategy.

2.4 Variation of time window

One of our aims is to determine how fraud detection performance changes as the length of the aggregation period varies. We vary the amount of information in the activity record, $\mathbf{y}$, by choosing time windows of different widths over which to accumulate transaction data. We may do this by considering different fixed numbers of transactions,

n in the argument to the transformation, $\Phi(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ above. Alternatively we may vary the width of a fixed time window (e.g. in terms of days). In the present work, the latter approach is adopted, but we will also consider single-transaction classification as a limiting case in which the time window is effectively zero (or $n = 1$). Many existing fraud detection systems only classify individual transactions, rather than accumulating account information. It will be interesting to quantify how much improvement in detection performance, if any, can be gained by considering more than one transaction in the context of 2-class classification.

In this study, we examine fixed time window periods of 1, 3 and 7 days respectively. For simplicity, these periods include the day of the current transaction and always start at midnight on the previous day, or 3 or 7 days ago. Note that if no transactions occur within a particular time window, then no activity record is generated, so a record is produced only when a transaction is observed. This can always be done, even if there is only one transaction in the window. We might have used a fixed number of transactions instead, but we feel that time periods are easier to understand from a practitioner's perspective. A fixed number of transactions might also entail a very long time period for certain accounts, which might be impractical. It is also important to note that ideally the aggregation period is a rolling window, so an activity record is updated with *every* transaction. This means that there would be a one-to-one correspondence between activity records and transactions, which allows the possibility of immediate fraud detection rather than having to wait for a fixed period of time to elapse before a new record is created. However, in practice we may have to be content with aggregating at particular times, such as once per day. The activity records in our study were generated at midnight each day (thus aggregating the previous n days of transactions). As noted above, we also compare the classifiers built using these time windows with classifiers built solely on transaction data (i.e. a window of length equal to 1 transaction).

One interesting possibility which we did not examine was to combine different aggregation periods within a single classifier. This would be one way of achieving a greater degree of 'scale invariance' among our input variables, as we could then see whether quantities such as 'average spend over last 3 days/average spend over last month (or year)' might be significant predictors. The use of such variables has a lot in common with the 'anomaly detection' (or behavioral model) approach. In practice, this strategy is likely to be advantageous, especially for longer aggregation periods, but we wanted to keep things simple and so examined different aggregation periods in isolation.

2.5 Classification methods

Although comparison of methods is not the primary aim of this study, we propose to use and compare several approaches to classification. By using a whole range of classification methods, we hope to be able to draw more general conclusions concerning fraud detection and transaction aggregation, rather than conclusions specific only to one or two particular methods. It must be remembered, though, that the optimal parameter settings for any method will vary from one data set to another. Where we

give parameter settings below, these were found to be good for our data and if we do not mention all parameters, then standard default values may be assumed. Two of the methods we used are representative of the more sophisticated recent approaches and have the property that they are capable (in their most advanced forms) of classifying any data:

- Support Vector Machines (SVMs) with Gaussian radial basis kernels (Cristianini and Shawe-Taylor 2000). The kernel width was set as the median distance (over all training examples) to the 40th nearest neighbor, as this gave good results on validation data. The regularization parameter, C, was set to 0.5 after experimentation with a wide range of values.
- Random Forests (Breiman 2001). This is a classifier in which a set of decision trees is generated on bootstrap samples of the data and then combined by majority voting (an example of a 'bagging' classifier). Unless otherwise stated, we used 200 trees in each forest (using more trees takes longer to train but did not improve the results). The 'mtry' parameter was set to 10 (no. of variables to consider at each splitting).

These will be compared with the following more traditional or 'simple' methods, which are also limited in terms of the kinds of decision boundaries that they can implement:

- Logistic Regression (see e.g. Hosmer and Lemeshow 2000), using all inputs.
- Quadratic Discriminant. Based on a two-class regularized multivariate Gaussian Bayes classifier (Friedman 1989).
- Naïve Bayes Classifier (see e.g. Hand and Yu 2001; Duda and Hart 1973). Each variable was divided into 50 intervals of equal frequency (i.e. the same total number of observations in each) where possible. The class-conditional probability density was then estimated in each interval. Variables with fewer than 50 values had one bin for each value. Note that all variables were effectively numerical (after aggregation).

Note that we used a Naïve Bayes classifier rather than a more general Bayesian Network, as the latter would be much more complicated to train and the former have been known to perform well despite violation of its independence assumption. The more general Bayesian Network should have similar capabilities to the Random Forest and SVM but if Naïve Bayes were found to perform equally well then its ease of use would make it a preferred choice. The same could be said of the other 'simple' classifiers.

In addition, the following methods are included as highly flexible universal classifiers, although they have been in use for many years and may be considered as 'standard' or 'traditional' approaches (for an overview see Hastie et al. 2001):

- Decision Tree (CART) (Breiman et al. 1984). The level of the tree was optimized using misclassification rate on validation data.
- K-nearest-neighbors (KNN). See, e.g. Hastie et al. (2001) for details. The fifth root transform was used to remove the otherwise excessive skewness of many of the variables. A Euclidean distance metric was applied after the transform and K was optimized using Gini coefficient on training data.

These approaches have been selected as they are either fairly common, or variants of commonly used classifiers, as well as representing a diversity of approaches with differing strengths and weaknesses. In addition, the fact that the authors have reasonable expertise in using all of these methods helps to provide a fair assessment between them. Practitioners in the banking sector rarely have special expertise in the use of these classification techniques, so usability is an important consideration. If a method requires a high degree of specialist knowledge and a great deal of time or effort in order to set the parameters, then it is less likely to be successful in practice (Hand 2006). This should be borne in mind when assessing the relative performance of these classifiers. Care was taken to invest roughly the same amount of human time and effort in the use of each of these classification techniques, with regard to optimizing parameters. Again, although such an assessment is subjective, it should help to make the comparison fairer. For this study, a decision was made to spend about one standard man-day of effort on each model. Generally, it is true to say that the performance of most classifiers can be improved in any given case by applying extra effort in fine tuning parameters or modifying the data. In this study, we made only rudimentary efforts in this regard and tried to ensure that no technique was favored. The reader should therefore be aware that the results reported here do not represent the very best performance achievable by each model with this data, although we believe they are close to optimal. In an ideal world, one would want to know the very best parameter settings for each method but this is not possible in practice. The procedure we followed is realistic and should provide a fair comparison, even though it does not necessarily produce the very best models.

2.6 Performance measurement

Most classifiers can produce a real number, monotonically related to the fraud probability, for each input pattern or record. This is referred to as the 'score', which is then compared with some threshold in order to produce a decision. Where we have reported 'scores' here, they are essentially fraud probabilities (under the assumption of equal priors) and are therefore directly comparable. By convention, a higher score is associated with higher fraud probability. If the decision threshold is set at a particular score, an alert is generated for any record with an equal or higher score. Records with lower scores are classified as non-fraud. Thus, only the ordering of the scores is important for performance measurement. All the measures we use are invariant to any monotonic transformation of score values. One measure of performance is simply the misclassification rate, although this takes no consideration of the varying costs of different types of misclassification. In addition, one should take care to ensure that the performance measure bears a close relation to the real costs incurred by banks faced with fraud. The actual industry average fraud rate on accounts is about 0.1% per annum in the UK (APACS 2006). This is reflected in the data we have, although we use sub-samples in which the fraud population is much higher, in order to develop classifiers. However, when calculating performance, we rebalance the fraud/non-fraud ratio to reflect the expected fraud prevalence of about 0.1%.

Performance measurement for fraud detection is a subtle matter, with many pitfalls and potential controversies. In particular, the Gini coefficient or AUC (area under the ROC curve) is often used as a standard measure of classification performance, although it has been pointed out that this may not be an appropriate measure for fraud applications (Adams and Hand 1999; Provost and Fawcett 1997). The reason for this is that the Gini or AUC is neutral regarding the relative cost of false negatives and false positives. In effect, these measures may be regarded as integrals over all possible cost ratios (Hand 2005). In order to evaluate fraud classifiers, one should really make some appropriate assumptions about the likely cost ratios involved and use these in a cost-weighted performance measure, such as the loss function described below, proposed in Hand et al. (2008). In the results section, we will focus on the loss function but we will also highlight where these differ from the Gini coefficient and why, arguing that the loss measure is more appropriate for fraud detection.

For this study, we assume that a false positive has a cost of 1 unit. This may be thought of as the cost of investigating a case. We will denote this as $c_{n/f}$ (the cost of a non-fraud classified as fraud). True positives will also incur this cost, denoted $c_{f/f}$. This is because an alert must be investigated before a fraud can be confirmed and there is always a cost associated with this. Hence $c_{n/f} = c_{f/f} = 1$. So, a fraud system produces alerts, which must then be scrutinized by a team of trained investigators. In most cases, an 'investigation' will consist of a brief look at recent transactions on an account, followed by a phone call to confirm that those transactions were made by the account holder. It may also require calls to the merchants involved, or more extensive work, depending on the size of the potential fraud loss. However, given that fraud is rare, there will always be a preponderance of false positives, so it is important to keep investigation costs down, especially when the potential loss is relatively small. Usually, a single phone call should suffice to ascertain the legitimacy of a set of transactions. The cost of the call is a few 10s of pence at most but the investigator's time is more expensive, together with overheads (office maintenance etc.), although fixed costs should be ignored. As a rough estimate, we will assume that a typical investigator can easily handle 50 investigations (essentially phone calls) per day and is paid about £400 per week. These figures were estimated with the help of our collaborating banks but they need not be perfectly accurate. We are only seeking some rough approximation of the true costs, which will vary widely across the industry. This implies a cost of about £2 per case, including the phone call. This may be compared with the typical cost of a known fraudulent transaction from the second data set described in Sect. 3.2. The mean amount of the fraud transactions was £140. This gives us a cost ratio of 70 but we will round up this number to 100. Formally, we will say $c_{f/n} = 100$, where $c_{f/n}$ is the cost of a fraud classified as non-fraud. This is equivalent to assuming a slightly more 'aggressive' fraud environment (where the cost of fraud is higher or its prevalence is increased). In any case, the assumed cost ratio is only a very approximate figure based on guesswork. In practice, any bank will have to make its own calculations based on its own costs. So, for this study, we will assume the cost of a false negative (i.e. undetected fraud) to be 100, as this is believed to be a reasonably realistic approximate figure. Finally, the cost of a true negative (non-alerted record), is assumed negligible and taken as zero. Thus $c_{n/n} = 0$.

For any given score and model, one may calculate the total cost associated with using that score as a decision threshold, as the sum of costs defined above for each record. However, the total cost may not be easy to interpret, so we will present 'normalized' costs, by dividing the total cost by the theoretical maximum cost, which is given by assuming that every record is misclassified. This then gives us a loss function, expressed as:

$$C(s) = \frac{c_{f/f} n_{f/f}(s) + c_{n/f} n_{n/f}(s) + c_{f/n} n_{f/n}(s) + c_{n/n} n_{n/n}(s)}{c_{f/n} n_f + c_{n/f} n_n} \tag{1}$$

Since $c_{n/n} = 0$ and $c_{n/f} = c_{f/f} = 1$ this is the same as the measure 'T1' proposed by Hand et al. (2008), but normalized to give a value between 0 and 1, allowing easy comparison between datasets of different sizes. Hence:

$$C(s) = \frac{n_{f/f}(s) + n_{n/f}(s) + c_{f/n} n_{f/n}(s)}{c_{f/n} n_f + n_n} \tag{2}$$

The cost in Eq. 2 is shown explicitly as a function of the score, s, which determines the numbers $n_{f/f}$ and so forth, representing the number of frauds classified as fraud. Thus, $n_{f/n}(s)$ is the number of fraud records classified as non-fraud (number of false negatives) when the threshold is s. The denominator is simply the maximum cost, where n_f is just the number of fraud records and n_n is the number of non-frauds. The optimum loss is simply the minimum of C(s) over all possible thresholds, s. It is this minimum value (on a validation sample) which we quote as the performance of a particular model. Note also that the one-to-one correspondence between activity records and transactions allows the same cost function to be applied regardless of the aggregation period.

Full details of how to calculate this cost performance measure are contained in Hand et al. (2008). Although it is much more relevant than a measure such as the Gini coefficient, it must be admitted that no performance measure fully captures all the practical aspects of fraud investigation. For example, it has been brought to our attention that banks will often avoid carrying out more than one investigation on a single account within a time period of a few weeks, in order to minimize inconvenience to the account holder. Clearly, this policy might lead to some frauds going unchallenged even though they may be detected by our system. We have decided not to take account of such policy decisions in our performance measure, since we would rather have a measure which is independent of the particular policies which individual banks may or may not apply. As a result, we believe our measure captures the essential common characteristics of the fraud intervention process and gives an idealized estimate of the potential benefit of the detection system under consideration, although it is possible that the full value of this benefit may not be realized in practice.

It must also be noted that we are assuming that our fraud classifier detection system will operate in addition to any existing anti-fraud processes used by the banks. Both of the banks in this study do have some basic rule-based fraud intervention system which leads to some transactions being queried. Some of the known fraud cases in our data have been detected after being queried by these systems; these accounts are

then terminated. Often, our classifiers can detect these cases even earlier (which leads to a positive benefit using our performance measure). However, in cases where our classifier does not detect the fraud, we cannot know how much fraud would have gone undiscovered, if the banks' existing systems did not exist. This data is censored by the existing anti-fraud process. Hence, any benefits we report here for fraud classifiers are additional to those obtained by the existing systems and supposes the continued use of those systems. Ultimately, though, many of the observed fraud cases are reported by customers after viewing their statements. This means that there is an upper time limit of about one month during which fraud can be carried out. The amount of the loss is also limited by the credit limit of the account.

3 Data

We describe results based on two independent data sets from two different banks. They each contain credit and debit card transactions, at both ATM and POS terminals, although we are focusing only on POS fraud in this study because we only had labeled data for POS fraud. Figure 1 is a frequency chart of how many transactions were made by each account in a 3-month period, but only up to 20 transactions (13% of goods had more than 20 transactions, along with 59% of the frauds but the shape of the distributions do not change). One can see that it is heavily skewed but that fraudulent accounts were less skewed, as these tended to be more active. In the first data set, fraud has been identified by specifying a date of first fraudulent transaction for each account where fraud has been committed. Thus, it is not possible to identify individual fraudulent transactions in this data set: one can only say that an account is in a fraudulent *state* at any given time. All records generated after the onset of fraud are labeled as fraudulent for training and validation. The second data set contained

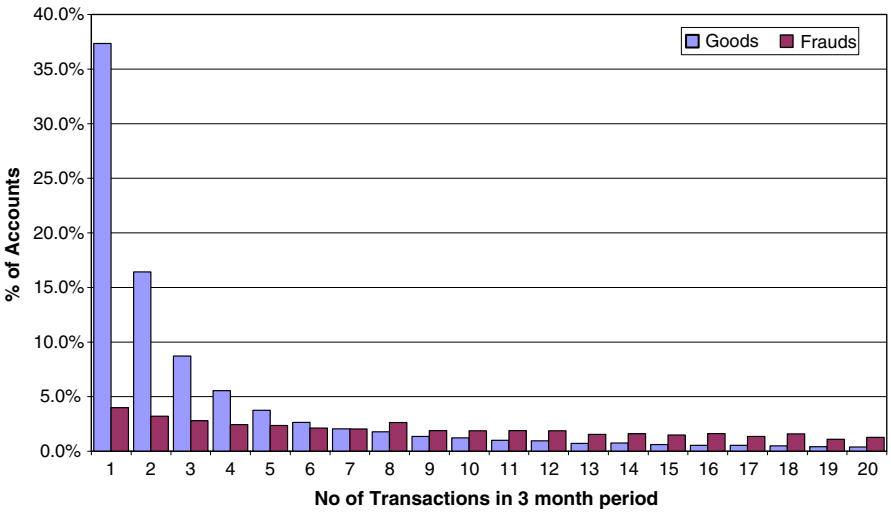

Fig. 1 Account activity

individually labeled fraud transactions. An account is considered to have entered a fraudulent *state* after the first fraudulent transaction has been committed. Hence, any record *summarizing* account behavior after that date must be considered fraudulent (i.e. labeled by S_t).

3.1 Bank A

This dataset contained 175 million transactions, collected between 1st August 2005 and 30th November 2005 with 76 fields per transaction. These belong to 16.8 million accounts, of which 5,946 experienced fraud (at POS terminals) during the observation period. Only the date of the first fraudulent transaction is given. For training and validation, the non-fraud accounts had to be considerably sub-sampled. Many accounts were inactive anyway and all accounts with no POS transactions were filtered out. All the 'fraud' accounts were retained as far as was feasible.

From this raw data, many fields were essentially uninformative (i.e. virtually single-valued). Only about 30 informative fields, deemed to have some relevance to the question of fraud, were retained. These fields were mainly categorical. Most of them were codes, indicating the following types of information:

- type of terminal
- transaction success status
- PIN use indicator
- checks made on the card (e.g. CVV code)
- merchant type
- transaction type (payment, deposit, transfer etc.)

One field also held the amount of each transaction. From these, further processing (as outlined in Sect. 2.1) produced 87 variables (some integer and some real-valued) for each activity record. These were mainly of two types:

- Counts of the number of transactions in each category (e.g. number of payments at a particular type of merchant) within a fixed time window
- Total value of transactions in each category (e.g. amount spent at terminals where card was not present) within a fixed time window

The transaction-level dataset (i.e. with no aggregation) contained only 45 variables, due to the fact that it is not necessary to count the number of transactions in each category. It is only necessary to supply the indicator variables and the amount field, as it is known that there is only one transaction. Hence, this data is inherently of lower dimension, with about half as many variables. In one way, this simplification is an advantage as it makes modeling easier. In another way, it may be disadvantageous, as the missing variables may contain useful information. This is partly what we are trying to ascertain.

Four complete datasets were produced, each with different time windows for aggregation. Transaction-level data was produced with a time window of effectively zero.

Activity records with windows of length 1, 3 and 7 days were also produced. It is important to note here that the parameters for each classification method were held constant with respect to the different aggregation periods (1–7 days), since these data sets were so similar. This helps to make performance comparison more objective, although parameters for the transaction-level models may be different since the dimensionality of this data set is smaller.

3.2 Bank B

This dataset was already highly filtered by the collaborating bank and contained 1.1 million transactions (75 fields each) observed between July 1, 2005 and December 31, 2005, pertaining to 21,655 accounts, of which 8,335 suffered some fraud.

The raw data was similar to that of bank A but not quite commensurable in every aspect, although there was a field containing the transaction amount, of course. After pre-processing (as for bank A), this resulted in activity records with 91 variables. Again, 4 full datasets were generated, using different time window periods (0, 1, 3 and 7 days as before), with transaction-level data equivalent to a period of length zero. The transaction-level data had only 47 fields, for the reason given earlier.

3.3 Training and test samples

In addition to the four different aggregation periods, the data were split into training and validation samples in 2 different ways:

1. *Prediction*: Records dated up to October 30, 2005 were used for classifier training. The classifiers were then tested on records dated November 1, 2005 onwards. This partition was chosen in order to simulate the effect of implementing a fraud classifier in a real environment, since fraud systems are required to work in situations where data distributions may be evolving over time. They are always required to extrapolate into the future, subject to the vagaries of population drift. Results will depend on how exactly the population has changed, so we cannot make safe generalizations from these.
2. *Random*: Records were assigned randomly (at account level) to training (70%) and test (30%) samples. This enables us to see if the models have over-fitted. As long as performance is roughly equal on the two samples, we may be confident that the models can generalize to 'unseen' data. This split may also be regarded as the purest test of the capabilities of the different models, in the absence of population drift.

In both cases, there was approximately a 70–30% split between training and test data respectively. In total, the aggregated datasets (with 1-, 3- and 7-day windows) contained about 46–48,000 activity records (training and test samples together). The transaction-level datasets were somewhat larger, containing around 60,000 records for bank A and 50,000 records for bank B.

3.4 Fraud rates

For bank A, the typical training data sample (using 1-, 3- and 7-day window sizes respectively) consisted of about 33–36,000 activity records, of which 33% were fraud. The transaction-only datasets (denoted 'tx' later) were slightly larger, with over 41–46,000 transaction records, of which 43–48% were fraud. The difference is due to the aggregation periods being full days ending at midnight, together with the nature of the fraud label. The validation samples contained over 11–14,000 activity records each, with about 30–33% fraud. For the transaction-only validation, there were 13–18,000 records, with 33–49% fraud. The variation reflects the fact that the fraud rate rose significantly over the observation period.

For the training sample of bank B, there were over 33–47,000 activity records and 35–37,000 transaction-only records, with about 36–50 and 27–36% fraud respectively. The validation samples contained 11–19,000 activity records, with 40–63% fraud. The transaction-only validation set had about 14–15,000 records, with 36–58% fraud. Again, the variation was due to changes in volume and fraud rate over the observation period.

Clearly, these fraud rates are not representative of the rates in the whole population, since our data are sampled to remove many of the accounts which were almost inactive (those with less than 3 transactions in total), as almost none of these experienced fraud. In addition the non-fraud accounts were randomly sampled to reduce the imbalance in class sizes, which makes model development much faster without sacrificing effectiveness. The dataset sizes were thus kept manageable whilst retaining as many fraud records as possible. However, as noted before, the populations were re-weighted back to the true expected proportions for the purpose of calculating cost-weighted misclassification rates. For this purpose, we took the 'standard' population fraud rate to be 0.1%, in line with our data as well as industry-wide estimates (APACS 2006).

4 Results and discussion

4.1 Prediction

Figure 2 shows the performance of all seven classifiers on the 'prediction split' of the data from bank A. Each group of four columns gives the minimum loss, as defined in Eq. 2, for each aggregation window. The leftmost column is for the transaction level classifier. The other columns give the minimum loss (over all decision thresholds) for aggregation windows of 1, 3 and 7 days from left to right. Figure 3 presents the performance on bank B, also for the 'prediction split'. There is a standard error (computed by bootstrap methods) for the loss function of roughly 0.001 in all these charts. The only method for which aggregation does not seem to work, with bank B data, is CART (see Fig. 3), where performance degrades as the period lengthens. This is discussed further in Sect. 4.3.

First, note that a cost of 0.091 (for both datasets) is achievable when no fraud system is employed (i.e. no alerts are generated at all) and fraudsters are simply allowed to carry out their activities. This performance level is shown by a dotted line in the

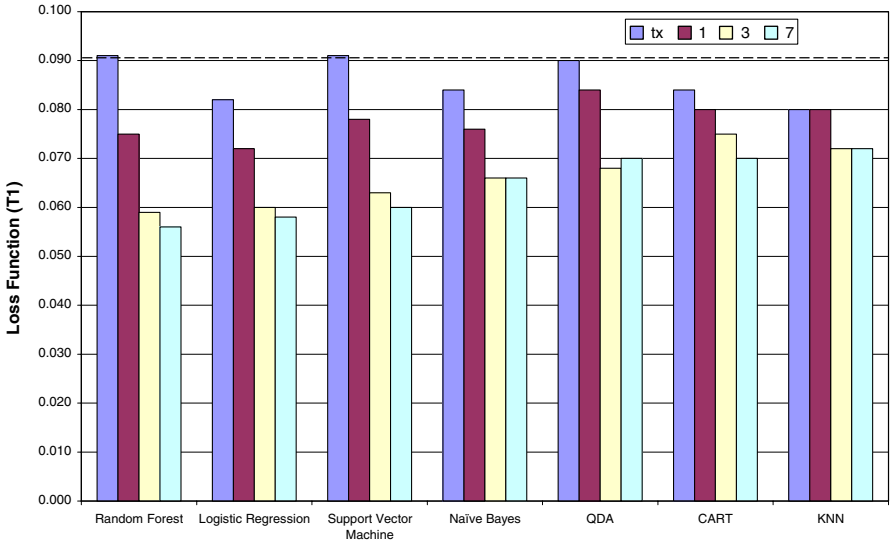

Fig. 2 Bank A prediction performance

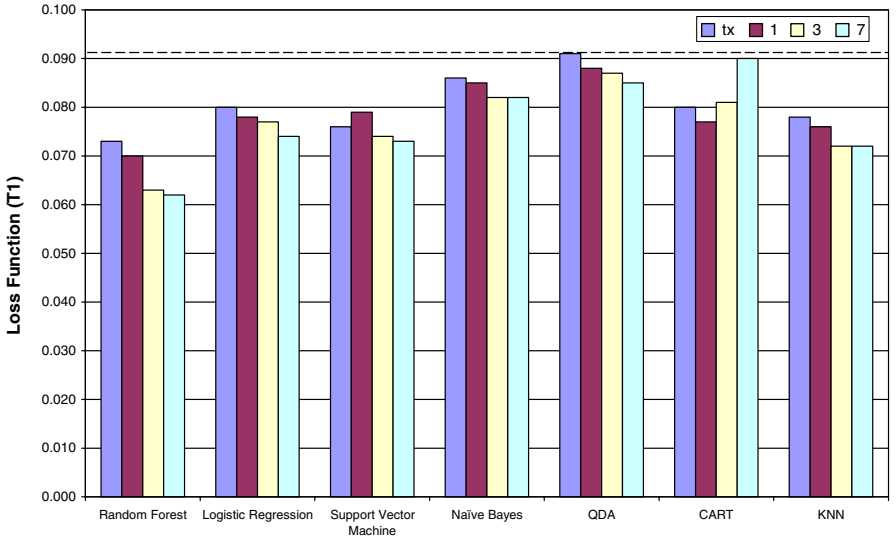

Fig. 3 Bank B prediction performance

charts. A genuine saving is only achieved when the loss drops below this level. One can see that several models do not achieve a useful performance, especially when transaction-only data ('tx') is employed.

The Random Forest appears to give the best results on both datasets, followed by Logistic Regression and SVMs, although the Random Forests and SVMs do not perform well on transaction-only data for bank A. For most (but not all) of the models,

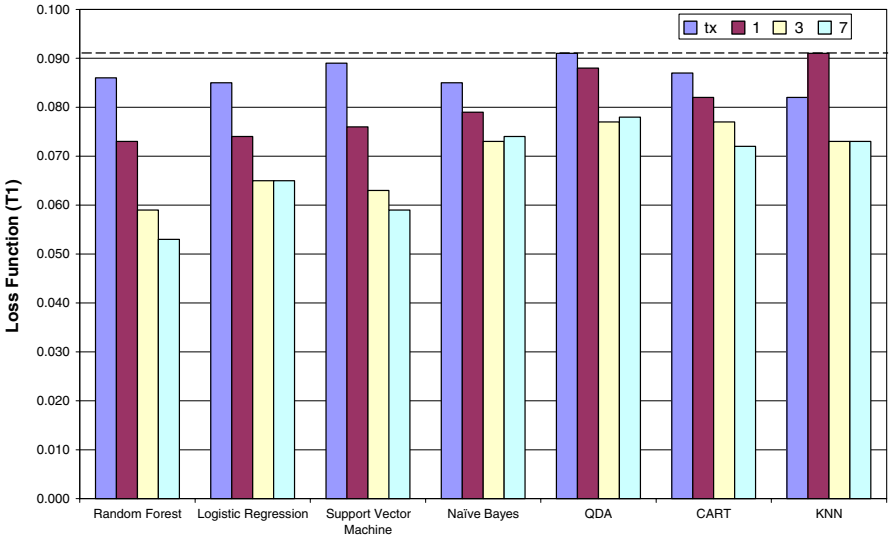

Fig. 4 Bank A random split performance

longer periods of aggregation give better results. This is especially true for the Random Forests, although it is hard to see a clear advantage in aggregating beyond 3 days. There does seem to be an advantage in using an aggregation period of longer than 1 day in nearly all cases, however.

4.2 Random assignment

Figures 4 and 5 present results on validation samples for the 'random split'. For bank A these results are very similar to those on prediction. For bank B, however, the transaction-level performance is much better than under the prediction split. The standard error is again around 0.001.

Although we have not shown the training data results, these are effectively identical to those on the test sample, differing by no more than 0.002 in every case. This confirms that good generalization has been achieved by the models. However, some models are clearly better than others. Again, the Random Forest stands out, especially at longer aggregation periods. In general, as the aggregation period grows, performance improves on bank A data. For bank B, the situation is less clear cut. By 3 days, the Random Forest is only slightly better than at transaction-level, but by 7 days there is clear improvement.

4.3 Population drift

For bank A, the two different data splits give very similar results, despite the inevitable effects of population drift over time (Kelly et al. 1999). For bank B, prediction performance at transaction level is not as good as it is for the random split. Transaction

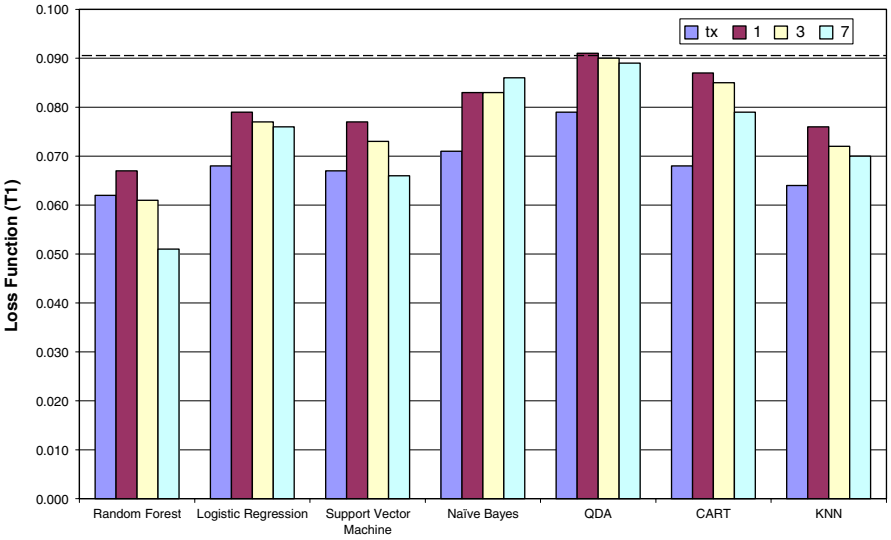

Fig. 5 Bank B random split performance

aggregation seems to be less affected by population drift, as the prediction performance is better for both banks with aggregated data than with transaction data.

The relative performance of the different classifiers does not seem to differ much between the two data splits. Population drift has a similar effect on most classifiers. However, Naïve Bayes does not seem to benefit from aggregation on the random split, whilst CART shows marked performance deterioration on the prediction split. It may be that CART in particular is more sensitive to population drift. This may be due to the tendency for a variable near the root of the tree to have a much greater influence on the results than variables nearer the leaves. Such a variable may work well on training data but could also suffer from population drift which will affect performance in the prediction scenario. The other classification methods give more equal consideration to the input variables, which ought to make them more robust in the event of population drift. In practice, this will be an important consideration, since population drift is inevitable.

For Naïve Bayes, it may be that performance is affected by the increasing correlations between variables as the aggregation period lengthens, as this leads to increasing violation of the independence assumption underlying the Naïve Bayes method.

4.4 Relative classifier performance

Random Forests emerge as a consistent winner in terms of performance in most circumstances, especially when using aggregated data. As a 'bagging' classifier, it does demand a fair amount of computing resources, similar to those required by SVMs, which also did very well, although finding the right parameters for the kernels also required some effort. Logistic regression also performed very well considering it was

one of the simplest methods to apply, requiring very little time or computing power. K-nearest-neighbors did particularly well on transaction-level data but not so well using aggregated records. This may be due to the considerable difference in dimensionality, as the aggregated records contain about twice as many variables and KNN classifiers are known to struggle in very high dimension. The worst classifier seems to be the QDA Bayes classifier. This could be because of the existence of near collinearities among the variables, in addition to excessive skewness, both of which tend to make covariance determination problematic, despite the use of regularization.

4.5 Score distributions and cost functions

The following tables give the means and standard deviations of the output scores (expressed as a probability of fraud under certain assumptions about priors) from the Random Forest models on Split 1 of the data (prediction) for bank B, broken down by the two classes, on transaction level and 7-day activity records respectively. The statistics in the first two columns are within-class, whilst the third column gives overall statistics regardless of class (Tables 3 and 4).

These tables show that the 'fraud-score' separation between the class means is greater for the transaction records, with a t-statistic of 2.3, compared to 1.7 for the activity records. However, the standard deviation of the fraud class is appreciably smaller for the activity records. This is a partial explanation for the better performance of activity records on the loss function measure (T1). A more detailed explanation is found when one looks at the empirical score (i.e. predicted fraud probability) distributions of the two classes at transaction level and activity record level respectively.

Figures 6 and 7 show the distributions of the non-fraud class ('diamonds' designated 'NF' in the figure legend) and fraud class (squares designated 'F') using essentially the same x-axis in each case. At transaction level, the non-frauds are skewed towards low scores but there is also a high proportion (about 5%) of frauds at this

Table 3 Transaction level score statistics by class

	Actual class		
	Non-fraud	Fraud	All
Mean score	0.111	0.711	0.325
SD of score	0.223	0.319	0.389
No. of observations	8856	4917	13773

Table 4 Activity record (7 day) score statistics by class

	Actual class		
	Non-fraud	Fraud	All
Mean score	0.434	0.788	0.606
SD of score	0.215	0.195	0.271
No. of observations	9818	9311	19129

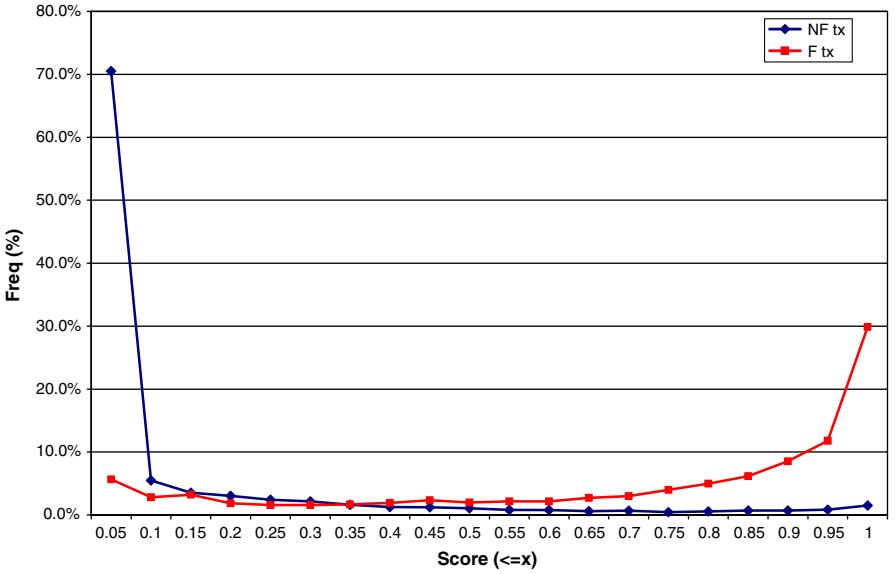

Fig. 6 Transaction empirical score distributions

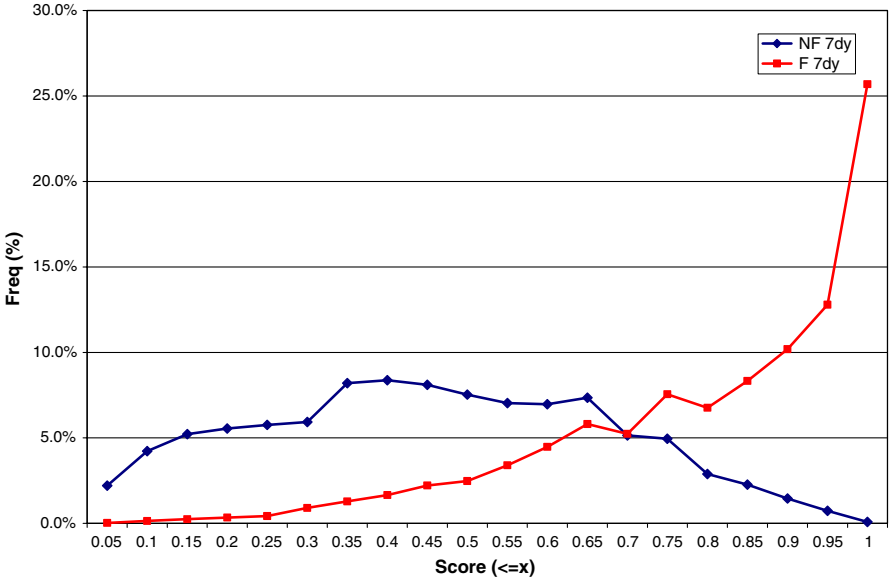

Fig. 7 Activity record empirical score distributions

end. By contrast, the activity record approach spreads the non-fraud observations more evenly across scores. However, the fraud observations are pushed up more away from the low scores, whilst the good observations are effectively bunched into the middle scores. This seems to give better discrimination in the regions where it matters most.

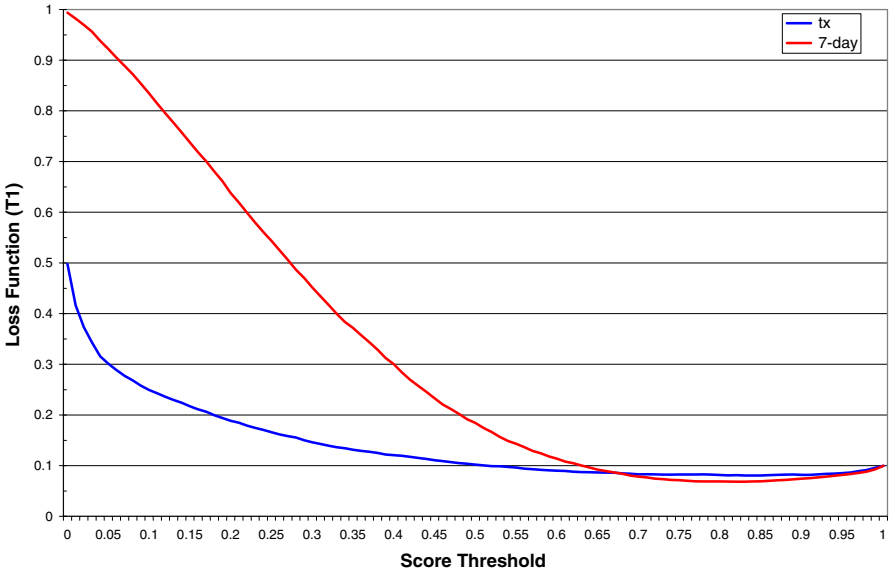

Fig. 8 Loss functions

It is interesting also to take a closer look at the cost-weighted loss (T1) as a function of score, s, as given by (2). Figure 8 shows T1 as a function of score threshold for the two models under discussion. Thus, the lines represent the total financial cost of investigating those alarms above the threshold, s, along with the cost of all frauds not found (below the threshold). The red line, representing 7-day activity records, is above the blue line, representing transaction records, until the threshold reaches 0.68. Only then does the activity record approach lead to a smaller cost-weighted loss. The improvement is important, however, and fully justifies the use of activity records for a cost ratio in the region of 100. Note that if no fraud system is deployed, the cost is 0.091. This level is reached at a threshold of 0.59 for transaction records and 0.66 for activity records. It may look as if there is not much improvement over having no system at all, but it must be remembered that the minimum loss of 0.062 represents a fraud cost reduction of about 32% (from 0.091) (Fig. 8).

The functions shown in Fig. 8 are the result of the class-conditional score distributions described in the preceding tables, which give some insight into why activity records work better than transaction records on loss function performance. By contrast, the Gini coefficient for the model based on transaction records is 85.0, compared to only 77.5 for activity records (with standard error ± 0.5). It is thus clear that the Gini index is an inappropriate measure for the performance of these systems (Hand 2005). One might wonder if other cost ratios would produce the same results. Our investigations confirm that this is so; for cost ratios varying between 25 and 125, activity records yield considerable improvement over transaction records. It is only at higher cost ratios that the difference is not so marked.

5 Conclusions

Before discussing our conclusions, it must be remarked that a study of this kind has certain limitations. Exact replication is not possible, although we suppose that many of the more general characteristics of our data would likely be reproduced in subsequent data, from different banks and different time periods. Fraud is, however, a notoriously fast-changing phenomenon, which responds to market conditions as well as to the measures taken by financial institutions against it. Any particular study is always a snapshot in time and space. Many of our conclusions will therefore be tentative. However, this is unavoidable if one wishes to study fraud. It is only by taking many studies together that one may arrive at a full picture. It should also be remembered that we have had to make some simplifying assumptions in the use of our cost function, which may not be exactly and universally applicable, although we believe they are sufficiently good approximations to the truth in most cases.

Probably the most important conclusion to draw from this work is that the aggregation period has a major impact upon the performance of classifiers for fraud detection. In the case of bank A, the different aggregation periods account for 88% of the variance in our performance measure in the prediction scenario and 78% of the variance in the randomized scenario. So, for bank A, the variation in performance due to the aggregation period is much greater than that due to the different classification methods. For bank B, the impact is smaller but still important, accounting for 29 and 43% of the variance in the prediction and randomized scenarios respectively. The lesson for practitioners is that they should pay at least as much attention and care to selecting appropriate aggregation periods as they do to selecting the best modeling techniques (and fine tuning their parameters).

So, aggregation is important but does it lead to better fraud classification? For bank A, this is clearly the case on the cost-weighted measure. This is unsurprising, as there is no class label information lost in using aggregated records (since we only have an approximate date of onset of fraud). For bank B the picture is less clear cut and more interesting. It seems that aggregation is only consistently better in the case of the Random Forest model, both for prediction and in the random data split (Figs. 3 and 5). Indeed, the Random Forest with aggregated data at 3 and 7 days produced clearly the best results with the data from both banks. This does seem to support the contention that data aggregation is an advantageous strategy, at least in some circumstances, although the advantage may be dependent upon the type of classifier and the amount of population drift. In particular, our study suggests that aggregation appears to work best under the following circumstances:

- With more ‘capable’ classifiers (Random Forests and SVMs in this case).
- When the fraud label is less precise (e.g. given by a date rather than for each transaction).
- Where population drift is a significant factor.

With regard to the last point, it should also be noted that aggregation produced improvements in the prediction scenario for all of the classifier types, except for CART (Fig. 3). This suggests that aggregation may be a more robust strategy for dealing with population drift.

These results are perhaps surprising when one considers that the bank B data contains labeled transactions, not just an indication of the fraud start date (as for bank A). This information is discarded when moving to the aggregation strategy. Yet, given a suitably long period of aggregation, the disadvantage of the missing class information can be overcome. Presumably, this is because useful new information is supplied in the form of transaction volumes or rates. Still, transaction-level classification does perform almost as well or better with most models for bank B on the random split (Fig. 5). This suggests that transaction-level classification should at least form a part of any successful overall fraud system.

The transaction-only approach is merely attempting to discriminate *known* fraudulent transactions from the rest, which may include both legitimate and possibly undiscovered fraud transactions. Indeed, it may be the case that a number of transactions labeled as non-fraud in this dataset are actually fraudulent. This might explain the elevated performance of the transaction-only classifier for bank B, if the known frauds are known precisely because they are particularly obvious.

The activity record approach is fundamentally different, in that it is attempting to classify an account as being in a fraudulent *state*, despite the possible presence of legitimate transactions. The decision does not attach to any individual transaction but rather to a whole series. This makes it particularly robust to the presence of mislabeled data, which is a common problem in the databases we are considering. Indeed, one does not require accurately labeled transaction data in order to apply this approach. It is only necessary to have historical data with approximate dates of onset of fraud. The good performance for transaction-only classification on bank B suggests that *confirmed* fraud transactions are relatively easy to discriminate. For example, the Gini coefficient at transaction level is 0.85 compared to 0.78 at 7-day level for Random Forests on bank B. However, this ease of classification does not translate into a great cost saving compared to the activity record approach. The reason for that may be better seen when one looks at a graph of the cost function over the range of possible score thresholds (see Fig. 6). As we have seen, the aggregated activity records give rise to a smaller dispersion for the score distribution of the fraud records (Table 4), although the two class means are better separated at transaction level (Table 3). This last fact also suggests a possible mechanism by which aggregation achieves easier classification: it seems likely that aggregation leads to smaller relative dispersion of input variables (as a consequence of the central limit theorem), especially within the fraud class. There is not sufficient space to prove this assertion here, but it could be the subject of a future study.

In summary, transaction aggregation is certainly useful in at least some situations and is probably essential when one does not have individually labeled transactions. Even when labeled transaction data is available, transaction aggregation may result in better performance. The appropriate aggregation period will probably vary from one dataset to another, depending on the typical timescale over which fraud takes place. The results here suggest that a period of at least 3 days is preferred for the datasets we examined, although it is not clear whether longer periods produce further advantage and it would be unwise to generalize on the basis of a single study. Another important point to note is that in practice, one might want to combine features based on very different aggregation periods, as this would allow the possibility of detection based

on differences in behavior over different periods. In the interests of simplicity, we did not examine this possibility but it would be interesting to devote some future study to this approach.

In practice, classification of aggregated records will be combined, whenever possible, with transaction-level classification and anomaly detection into a complete fraud detection system. One would hope that information could be combined from all 3 subsystems in such a way as to produce much better fraud detection than any single sub-system could achieve on its own. For example, if an anomaly detector produces an alert that behavior for an account has changed, this is much more likely to be due to fraud if a fraud classifier also produces an increased fraud score for that account. A transaction level classifier would lend extra support to this conclusion if it could identify several individual transactions on that account which look fraudulent. Future work will try to address the question of how best to combine the outputs of these three sub-systems.

Acknowledgements The work of Piotr Juszczak and David Weston described here was supported by the EPSRC under grant number EP/C532589/1: *ThinkCrime: Statistical and machine learning tools for plastic card and other personal fraud detection*. The work of Chris Whitrow was supported by a grant from the Institute for Mathematical Sciences, Imperial College London. The work of David Hand was partially supported by a Royal Society Wolfson research merit award. We are indebted to our commercial collaborators who prefer to remain anonymous for providing both data and useful insights about the fraud detection problem.

References

Adams NM, Hand DJ (1999) Comparing classifiers when the misallocation costs are uncertain. Pattern Recognit 32:1139–1147. doi:10.1016/S0031-3203(98)00154-X

Aleskerov E, Freisleben B, Rao B (1997) CARDWATCH: a neural network based database mining system for credit card fraud detection. In: Computational intelligence for financial engineering. Proceedings of the IEEE/IAFE. IEEE, Piscataway, NJ, pp 220–226

APACS (2006) Fraud The Facts 2006. Retrieved May 16 2007, http://www.apacs.org.uk/resources_publications/apacs_publications_2.html

Bolton RJ, Hand DJ (2001) Unsupervised profiling methods for fraud detection. In: Conference on credit scoring and credit control, vol 7. Edinburgh

Bolton RJ, Hand DJ (2002) Statistical fraud detection: a review. Stat Sci 17:235–249. doi:10.1214/ss/1042727940

Brause R, Langsdorf T, Hepp M (1999) Neural data mining for credit card fraud detection. In: Proceedings of the 11th IEEE international conference on tools with artificial intelligence, pp 103–106

Breiman L (2001) Random forests. Mach Learn 45:5–32. doi:10.1023/A:1010933404324

Breiman L, Friedman JH, Ohlsen RA, Stone CJ (1984) Classification and regression trees. Wadsworth, Belmont, CA

Chan PK, Fan W, Prodromidis AL, Stolfo SJ (1999) Distributed data mining in credit card fraud detection. Intell Syst Their Appl IEEE 14:67–74. doi:10.1109/5254.809570

Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines. Cambridge University Press, Cambridge, UK

Dorronsoro JR, Ginel F, Sanchez C, Santa Cruz C (1997) Neural fraud detection in credit card operations. IEEE Trans Neural Netw 8:827–834. doi:10.1109/72.595879

Duda RO, Hart PE (1973) Pattern classification and scene analysis. Wiley, New York, NY, pp 10–43

Fair I (2007) Falcon^{TM} Fraud Manager web page. Retrieved June 14, 2007, http://www.fairisaac.com/fic/en/product-service/product-index/falcon-fraud-manager/

Fawcett T, Provost F (1997) Adaptive fraud detection. Data Min Knowl Discov 1(3):291–316

Friedman JH (1989) Regularized discriminant analysis. J Am Stat Assoc 84:165–175. doi:10.2307/2289860

Ghosh S, Reilly DL (1994) Credit card fraud detection with a neural-network. In: Nunamaker JF, Sprague RH (eds) Proceedings of the 27th annual Hawaii international conference on system science vol 3: information systems: DSS/knowledge-based systems. Los Alamitos, CA, USA

Hand DJ (2005) Good practice in retail credit scorecard assessment. J Oper Res Soc 56:1109–1117. doi: 10.1057/palgrave.jors.2601932

Hand DJ (2006) Classifier technology and the illusion of progress (with discussion). Stat Sci 21:1–34. doi:10.1214/088342306000000060

Hand DJ, Yu K (2001) Idiot's Bayes—not so stupid after all. Int Stat Rev 69:385–398

Hand DJ, Whitrow C, Adams NM, Juszczak P, Weston D (2008) Performance criteria for plastic card fraud detection tools. J Oper Res Soc 59:956–962. doi:10.1057/palgrave.jors.2602418

Hastie T, Tibshirani R, Friedman JH (2001) Elements of statistical learning. Springer

Hosmer DW, Lemeshow S (2000) Applied logistic regression. Wiley

Kelly MG, Hand DJ, Adams NM (1999) The impact of changing populations on classifier performance. In: Chaudhuri S, Madigan D (eds) Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining. Association for Computing Machinery, New York, pp 367–371

Kou Y, Chang-Tien L, Sirwongwattana S, Huang Y-P (2004) Survey of fraud detection techniques. In: IEEE international conference on networking, sensing and control, pp 749–754

Maes S, Tuyls K, Vanschoenwinkel B, Manderick B (2002) Credit card fraud detection using Bayesian and neural networks. In: Proceedings of first international NAISO congress on neuro fuzzy technologies: NF2002, Havana, Cuba. NAISO Academic Press, Canada/The Netherlands, pp 16–19

Provost F (2002) Comment on Bolton and Hand (2002). Stat Sci 17:249–251

Provost F, Fawcett T (1997) Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. In: Proceedings of the third international conference on knowledge discovery and data mining, pp 43–48

Wheeler R, Aitken S (2000) Multiple algorithms for fraud detection. Knowl Base Syst 13:93–99. doi: 10.1016/S0950-7051(00)00050-2