A data mining based system for credit-card fraud detection in e-tail

Nuno Carneiro, Gonçalo Figueira*

INESC TEC and Faculty of Engineering of the University of Porto

Miguel Costa

Mathematical Institute, University of Oxford

Abstract

Credit-card fraud leads to billions of dollars in losses for online merchants. With the development of machine learning algorithms, researchers have been finding increasingly sophisticated ways to detect fraud, but practical implementations in are rarely reported. We describe the development and deployment of a fraud detection system in a large e-tail merchant. The paper explores the combination of manual and automatic classification, gives insights into the complete development process and compares different machine learning methods. The paper can thus help researchers and practitioners to design and implement data mining based systems for fraud detection or similar problems.

Keywords: Fraud detection, Credit-Card, Online retail, Supervised learning, Practical implementation
2010 MSC: 00-01, 99-00

1. Introduction

Fraud is a major problem for merchants, particularly in the online sector. The total revenue loss for e-commerce merchants in North America amounted to \$3.5bi in 2012, which represents 0.9% of the total revenue. For orders outside of North America, the fraud rate was almost twice as high, averaging 1.6% [1].

[2] define credit-card fraud as the action of an individual who uses a credit-card for personal reasons without the consent of the owner of the credit-card and with no intention of repaying the purchase made. Contrary to what many consumers believe, merchants are responsible for paying the bill when a fraudster steals goods or services in a consumer-not-present transaction, such as an online payment. A *chargeback* occurs when a consumer claims that he/she did not get the products or services requested, or that the order was placed by a fraudster. If the company cannot rebut this claim, the money will have to be returned to the consumer's account and the product is lost (if it has been shipped). Moreover, merchants can be subject to chargeback fees and fines from card associations if the chargeback rate is above their thresholds [3].

Several technologies have been used to prevent fraud from happening, such as the Address Verification System (AVS), Chip and Pin verification and Card Verification Code (CVV). However, even these advanced systems are prune to fail. The development of fraud detection methods is thus of crucial importance.

Fraud detection is a challenging problem because fraudsters make their best efforts to make their behaviour look legitimate. Another difficulty is that the number of legitimate records is far greater than the number of fraudulent cases. Such unbalanced sets require additional precautions from the data analyst. [4] claim that the key for accurate fraud detection lies in developing dynamic systems that can adapt to new fraud patterns. Fraud detection must therefore evolve continuously, faster than fraudsters.

This problem has been addressed in different contexts (credit-card transactions, credit application, telecommunications) with multiple approaches [3]:

*Corresponding author
Email address: goncalo.figueira@fe.up.pt (Gonçalo Figueira)

- identity proofing, which has the aim of verifying the customer's identity using publicly available data or making a phone call;
- guaranteed payments, such as merchandise insurance or authentication services, provided by card associations in exchange for a fee;
- expert-rules, including all the techniques based on rules created from the experience of fraud analysts (e.g. positive and negative lists of users);
- data sharing, which consists in acquiring information about the customer from an external provider or social network monitoring;
- technology services, such as biometrics systems or the detection of the use of proxy servers;
- and data mining techniques that take into account past transactions to estimate the probability of a new transaction being fraudulent.

Some of these approaches involve obtaining additional information (at a given cost), while others make use of the existing data. The latter can be divided in two main categories: manual (such as identity proofing) and automatic (mostly data mining). When large volumes of data are involved, pure manual detection becomes impractical. Therefore, data mining techniques are imperative. Many authors have explored these techniques with data from banking and credit-card operations ([5], [6], [7], [4], [8], [9], [10], [11]), with a focus on supervised methods, such as Artificial Neural Networks, Support Vector Machines and Logistic Regression. To the best of our knowledge, only one study [12] has examined this problem from the perspective of an e-commerce merchant. An important difference to the banking sector is that a single online retailer has very limited information about the customer it is doing business with. Moreover, few authors have dealt with the discussion of practical aspects such as which data to use. [13] point out that only seven studies had claimed to have been implemented in practice by 2010. Since then, we found two more practical implementations. However, none of them has considered the combination of data mining and manual revision in the whole system. Indeed, [14] conclude that the most urgent challenge facing fraud detection is "to bridge the gap between practitioners and researchers", and that "research should direct its attention toward finding more practical principles and solutions for practitioners to help them to design, develop, and implement data mining and business intelligence systems".

This paper reports the development and implementation of a fraud detection system in a large e-tail merchant (for confidentiality reasons the name of the company will not be disclosed). The system explores the combination of manual and automatic classification, based on real data from the company case study. The final system is evaluated and compared to current practice. Our work thus contributes to the literature in the following way:

- providing a case study from e-tail, which has important differences when compared to the well studied banking sector;
- exploring the combination of an automatic classifier with manual revision;
- giving a practical perspective on the complete process of developing such a solution, including how to select, collect, label and transform data;
- showing a comparison of different supervised learning methods;
- describing the practical deployment at an e-tail company.

We followed the methodology suggested by [15]. At first, we gained understanding of the key business concepts and objectives involved. We had interviews with the involved parties at the company and shadowed the fraud analysts at work. The second step involved building the data set of orders history. An exploratory statistical analysis followed, where we identified possible patterns of fraud by looking at individual variable distributions. The outcome was a report which could be presented to the fraud analysts to guide them in their revision process. The third step consisted in data preparation. The variables which could not be included in the deployed model were left out; the rest was transformed and standardized, to be used by machine learning algorithms. We have trained, tested, parameterized and compared three distinct algorithms: Logistic Regression, Random Forests and Support Vector Machines. Lastly, we have evaluated the results and deployed the solution at the company.

The remainder of this document is organised in six more sections. Section 2 introduces the case study and the proposed approach. Section 3 presents an overview of the data mining methods for fraud detection found in the literature, including supervised and unsupervised learning approaches, giving context to our selection of techniques to test. Section 4 details the many steps taken to collect and prepare the data. Section 5 discusses the different models, their evaluation and cross-validation, providing already an estimation of the final performance. Section 6 explains the challenges in deploying the solution and the first results. Finally, Section 7 presents conclusions and suggestions for future work.

2. Fraud detection case study

The company of this case study is one of the leading online luxury fashion retailers in 2015. It follows a marketplace business model, selling items from more than 400 partner boutiques on a commission basis. The value proposition for boutiques includes payment processing, branding, online content creation, out-bound logistics and customer service. As part of payment processing, fraud detection is included as an added value service provided. Moreover, being acknowledged by customers and card associations as a trustworthy merchant is crucial when doing business online.

Fraud detection involves a fundamental trade-off. On the one hand, the company has to minimize the level of fraud, maximizing the detection of fraudulent transactions, thus avoiding chargebacks. On the other hand, it must provide high payment acceptance rates in order to convert as many sales as possible and minimize the number of customer insults (i.e. the number of legitimate transactions refused). At the moment, the company relies on expert-rules to perform a first check, but the majority of the orders (around 65%) are manually verified by the order approval team for any indications of fraud. This is time consuming and is not sustainable, as the company's prospects for the mid-term future foresee a continued growth rate around 50-70%/year. Increasing the automation level is thus essential to quickly evaluate a large and growing number of orders. This high growth rate introduces some nuances to the problem, since the history of orders to study from is skewed towards more recent dates. In addition, this growth made the company a more popular target for fraud. Moreover, the company ships to everywhere in the world, which increases the difficulty of preventing fraud.

The company tracks several key performance indicators (KPIs) related to fraud detection in daily, weekly and monthly reports. The main KPIs consist of:

- the automation level – percentage of orders automatically processed;
- chargeback level – percentage of orders which originate a chargeback;
- rate of refused payments – percentage of payments which are refused;
- and speed of processing – time it takes to approve or reject an order payment.

The objective of the company is to develop a system which will result in a higher automation level, while not increasing the chargeback level and the rate of refused payments. In the medium-term the new process should allow the company to reach a level of automation of 80%, with a chargeback level under 1% and a payment refused rate under 4.5%.

The proposed approach consists in building a risk scoring system based on machine learning methods which will estimate a *Fraud Suspicion Score* for each order. A diagram of the proposed process can be seen in Figure 1. The suspicion score estimated by the model should be a number between 0 and 1. The risk scoring model would also evaluate whether the score falls below a certain threshold (e.g. lower than 30%), where the order would be automatically approved; or higher than an upper threshold (e.g. higher than 70%), where the order would be automatically rejected. Orders with a score between the lower and the upper thresholds would be manually revised by the order processing team, which could also count on the suspicion score for a better evaluation.

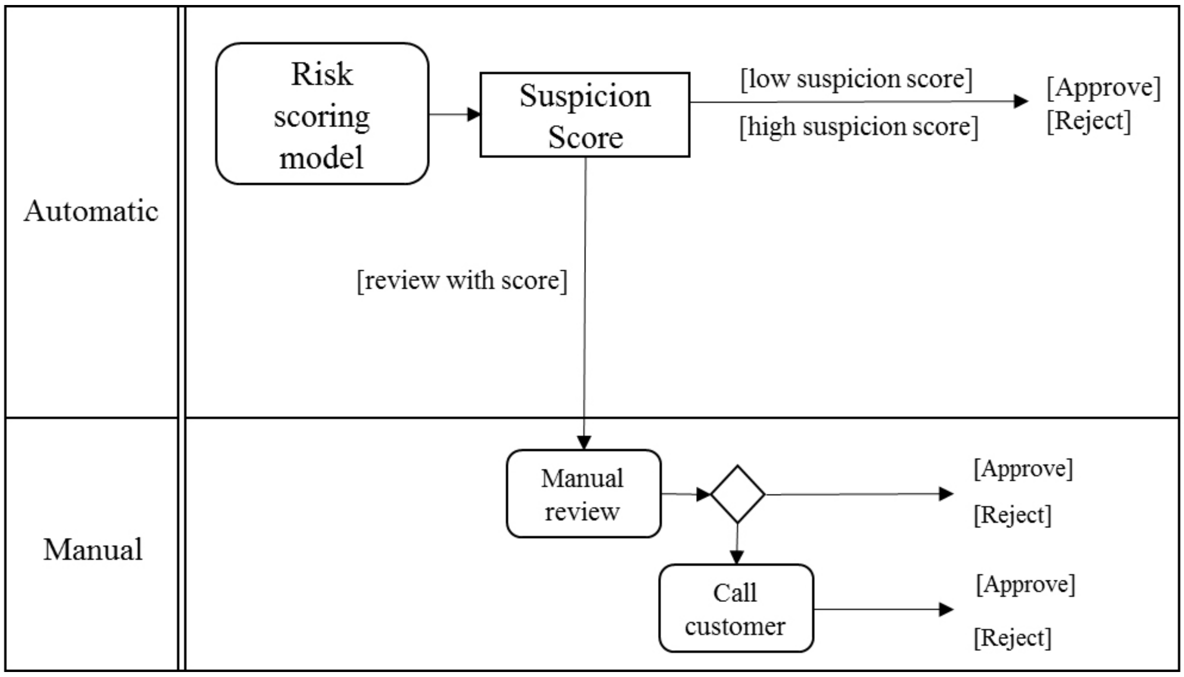


Figure 1: Proposed process for fraud detection at the company

3. Data mining techniques for fraud detection

According to [13], the most cost effective approach for fraud detection is to "tease out possible evidences of fraud from the available data using mathematical algorithms". Data mining techniques, which make use of advanced statistical methods, are divided in two main approaches: supervised and unsupervised methods. Both of these approaches are based on training an algorithm with a record of observations from the past. Supervised methods require that each of those observations used for learning has a label about which class it belongs to. In the context of fraud detection, this means that for each observation we know if it belongs to the class "fraudulent" or to the class "legitimate". Often we do not know which class an observation belongs to. For example, take the case of an online order whose payment was rejected. One will never know whether this was a legitimate order or whether it had been correctly rejected. Such occurrences favour the use of unsupervised methods, which do not require data to be labelled. These methods look for extreme data occurrences or outliers. In order to get the best of two worlds, some solutions combine supervised and unsupervised techniques.

A few authors have studied unsupervised methods for fraud detection. [16] explored the use of graph analysis for fraud detection in a telecommunications setting. [4] proposed a mixed approach with the use of a self-organising map which feeds a Neural Network if a transaction does not fall into an identified normal behaviour for the given customer. [17] compared supervised and unsupervised Neural Networks. According to their experiment the unsupervised method performed far below the supervised one.

Supervised methods have dominated the fraud detection literature. In general, the emphasis of research in the late 90s and early 2000s was on Neural Networks. [5] proposed the use of a Neural Network for fraud detection at a commercial bank. [6] studied the use of a profiling approach to telecommunications fraud. [18] discussed the combination of multiple classifiers in an attempt to create scalable systems which would be able to deal with large volumes of data. More recently, some other works have been published, making use of newer classification techniques. [7] built a model based on a Hidden Markov Model, with focus on fraud detection for credit-card issuing banks. [8] also worked on credit-card fraud detection with data from a bank, in particular addressing the way of pre-processing the data. They studied the use of aggregation of transactions when using Random Forests, Support Vector Machines, Logistic Regression and K-Nearest Neighbour techniques. [10] compared the performance of Random Forests, Support Vector Machines and Logistic Regression for detecting fraud of credit-card transactions in an international financial institution. Random Forest proved to be the most effective and most versatile method in this case.

[13] pinpoint two criticisms to the data mining studies of fraud detection: the lack of publicly available data and the lack of published literature on the topic. Most literature on credit-card fraud detection has focused on classification models with data from banks. Such data invariably consists of transaction registries, where it is possible to find fraud evidence such as "collision" or "high velocity" events, i.e. transactions happening at the same time in different locations. Some authors have also addressed the techniques for finding the best derived features. [8] proved that transaction aggregation improved performance in some situations, with the aggregation period being an important parameter. However, none of these particularities seems to apply to a case of detecting fraud with data from one single merchant as in our case.

In this study, we chose to use methods of supervised learning for the classification problem, because it is common for fraud detection applications to have labelled data for training. We chose to test three different models. Logistic regression because of its popularity, and Random Forests and Support Vector Machines, which have been used in a variety of applications showing superior performance [10]. [19] showed that Support Vector Machines perform well in classification problems. [10] claim that Random Forests are very attractive for fraud detection due to the ease of application and being computationally efficient.

4. Data understanding and preparation

Building the dataset on which to base the study is not a trivial activity and requires decisions which can greatly affect the quality of the data mining project. Which data to use? Where is this data to be found? Moreover, one must transform all categorical variables into numerical, in order to use machine learning algorithms such as Support Vector Machines. This section describes the reasoning behind the building of the data set which was the basis of this study. The handling of the data was done with the use of the programming language Python [20] version 2.7, available through the Python Software Foundation at http://www.python.org. Two particular modules for Python, Pandas data structures [21] and Scikit-Learn [22], were especially useful for running the algorithms.

4.1. Data collection

Taking into consideration the input from the fraud analysts at the case study company and the examples in the literature, we decided that the unit of analysis would be each individual order. With that in mind, the objective at this stage was to build a data set where each row would correspond to one order, the columns would represent different attributes of such order. The bulk of data was queried from a table where each line describes one product item in an order. Aggregating multiple products involved creating additional attributes for the order. For instance, a column was created for the number of products corresponding to each gender ("Men", "Women" or "Other"). The same was done for the product family ("Clothing", "Shoes", etc.). Product brands could not be done in the same way, because there were more than 2500 possible brands. Therefore, we chose to create individual columns for the 20 best-selling brands only. Other products attributes such as item price or quantity were simply summed. Building the data set and aggregating data involves a trade-off. A higher level of aggregation results in a larger (more significant) number of occurrences per category. On the other hand, the loss of information can be substantial if the aggregation level is too high. Therefore, a balance had to be found. Additionally, we merged data about the payment processing (e.g. AVS, CVV responses) and the customer's behaviour online before the purchase (e.g. number of pageviews, etc.).

Another key aspect in classification is the definition of the class for each data entry. In our case, we faced a binary classification problem where each data entry – an order – has to be labelled as belonging to one of two classes: fraudulent or non-fraudulent. Two criteria are taken into consideration when defining the classes of the dataset and we created a variable which is called *LabelFraud* with a value of 0 or 1, such that:

1. An order that originated a chargeback or was manually marked as fraudulent (see Figure 2) was labelled as fraudulent (*LabelFraud* = 1);
2. All other orders were labelled as legitimate transactions (*LabelFraud* = 0).

Not all data can be labelled, due to the long time it takes for the merchant to realise that a fraudulent transaction has occurred. Most commonly, the alert is first raised by the legitimate owner of the credit-card, who notices a transaction in his bank statement that he did not make. Then, he must place a complaint on that transaction, which will originate a

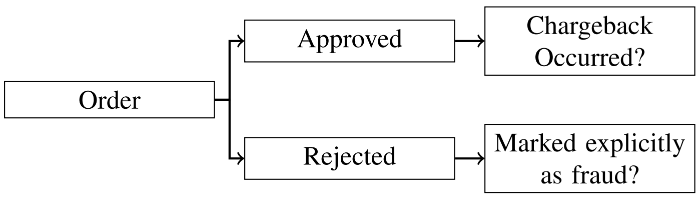


Figure 2: Criteria for labelling an order as fraudulent.

chargeback. It seems natural that such lengthy procedures take several weeks between the time that the fraudsters use the credit-card and the merchant is notified of the chargeback. This restricts the orders we can use for data mining, because the most recent frauds will not have been detected yet. In the past, 80% of the chargebacks arrived within the first 11 weeks after the transaction. Knowing this, it was decided that the data to be analysed should be no more recent than 16 weeks (circa 4 months), this criterion was also included in order not to take the most recent data into consideration when training the algorithm in the future.

The collected data is split in two: a training and a testing set. The training set is then used for the algorithm to learn how to classify orders. The testing set is used just at the end to validate the effectiveness of the algorithm, removing the overfitting effect, i.e. the increase in performance that the algorithm has on the instances it has based its learning, as it describes not only the underlying relationship, but also random error (present in those particular instances).

According to the literature, the division between the number of observations in the testing set should be between 20-40% of the total number of data entries. We chose to consider the most recent 20% of orders for testing. Dividing by date was based on the idea that patterns of fraud change over time and more recent orders would better emulate new orders. The result of the division is a training set containing the orders placed between 2015-01-01 and 2015-06-21 (totalling 347,572 observations), and a testing set containing the orders placed between 2015-06-22 and 2015-08-04 (in a total of 86,893 orders).

4.2. Variables transformations

Categorical variables with few categories were transformed into several features through one-hot-encoding. One-hot-encoding is a method which consists of encoding a variable through binary numbers. In our case, we created one column for each of the categories of the variable. This technique will allow the model to fit such variables, but as the dimensionality grows the model looses generality [23]. Hence, one-hot-encoding should not be applied to variables with an extensive number of categories.

Variables such as the address country include too many categories (more than 200 different countries) to be transformed into individual features through one-hot-encoding. Therefore, we chose to cluster countries by fraud risk, calculated by the ratio of fraudulent orders over the total number of orders in that country *i*:

$$FraudRatio_i = \frac{FraudulentOrders_i}{TotalOrders_i} \tag{1}$$

The countries have been clustered in four groups. Countries with less than 30 orders, have been considered not significant and attributed an intermediate level of risk. When transforming an entry for a new order, we check which risk-level had been assigned to that country in the training phase. Had such a category not been assigned a risk-level before (e.g. a country where no order from the training set was shipped to), the risk-level is set to the intermediate level.

In order to get more significant variables to train the models, we engineered new variables through abstraction and combination of variables. The first engineered variables were created to represent the degree of similarity between certain pairs of categorical variables. We created a binary function which generated a new feature with the value of 1 in case of a match and 0 in case of no match. This function was applied to the pairs of variables {(billing country, shipping country), (billing country, card country), (shipping country, card country)}. These were all predefined country fields, which ensured the match could be done by a simple comparison of strings.

On the other hand, the variables referring to names (e.g. name on card, user name) would often not match because the names were spelled in different order. Hence, we used another function which calculates the similarity between two names. This function is based on n-gram similarity and its output is a continuous number in [0,1]. It was applied to the pairs of variables {(UserName, CardName), (billing city, shipping city), (billing zip-code, shipping zip-code)}. A function which confirms that the customer has entered a valid telephone number was also created, by checking whether the input of the user is a number. Lastly, a function summarizing the customer's recent buying behaviour was created. This function counts the number of orders by this customer in the ten days prior to the current purchase.

4.3. Variables standardization

In order to get the best performance of the machine learning algorithms, the data must be clean and complete. We chose to complete missing data entries by imputing a value in the missing variables, as suggested by [23]. In the case of categorical variables, a missing value would correspond to a zero in each of the corresponding columns after the one-hot-encoding transformation. In the case of numerical variables, the missing value was replaced by the mean of the sample of the training set.

Support Vector Machines and other machine learning methods require data variables to be in the same range. In our case, variables which are measured in different units such as *Quantity* or *OrderValue* were standardized to values between 0 and 1, so that they can more easily be compared. Among the several techniques for standardization, we chose to use Min-Max. This technique was preferred over the commonly used Z-value standardization, because it transforms variables to the range [0,1], which is consistent with the range of the many binary variables in the model.

5. Modelling and evaluation

This section describes the selection of the model and parameters to use for the order classification. Subsection 5.1 introduces the measures used to evaluate the performance of the different algorithms. Subsection 5.2 details the evaluation of the different models: Logistic Regression, Random Forests and Support Vector Machines. The choice of the models and respective parameters was done by applying cross-validation on the training set. The results of testing with the chosen model are discussed in Subsection 5.3. An automatic approach to defining the score threshold is considered in Subsection 5.4. Subsection 5.5 elaborates on a second approach to defining the threshold, based on the restriction of the number of orders which can be manually revised.

5.1. Performance measures

Before evaluating the results of each model, we had to decide what would be the performance measures to compare. For each observation X, we have an associated real class label from the set {**0,1**} and a corresponding predicted label. Observations which are classified correctly as belonging to class 1 are called *true positives*, while observations correctly classified as class 0 are called *true negatives*. There are two other possible outcomes in which the prediction incurs in an error. The *type I error* occurs when the positive class is predicted, but the observation label is 0 (these predictions are called *false positives*). The *type II error* occurs when the prediction of class 0 does not agree with the observation's true class which would be 1 (*false negatives*). The two types of error can have different implications. In the case of fraud detection, a false positive error would correspond to a legitimate observation being labelled as fraud. A false negative error would happen in the case of a fraudulent transaction being classified as legitimate.

In the case of classifiers such as Random Forests, the output is a continuous value, that represents a score which tells us how close the observation is to 0 or 1. Therefore, a threshold must be defined in order to determine the final class (0 or 1), based on the real value obtained with the classifier. The *Receiver Operating Characteristic* curve (ROC) offers a way of visualizing different outcomes. [24] describes ROC curves as depicting the relative trade-offs between benefits (true positives) and costs (false positives), working very well in practice as a general measure of classifier performance. Observations with a score under the threshold are classified as class 0, while a score above the threshold would predict that the observation belongs to class 1. A ROC curve plots the *true positive rate* and *false positive rate* for each threshold between $-\infty$ and $+\infty$ [24]. The area under the curve (AUC) is equivalent to the probability that the classifier will give a higher score to a randomly chosen observation of class 1 than to a randomly chosen observation of class 0 [25].

Precision-Recall (PR) curves have been used as an alternative performance measure to ROC. In a PR graph, we plot the precision and recall for each threshold. [26] studied the relationship between PR and ROC curves. One of the conclusion of this study is that optimizing the AUC-ROC will not guarantee the best result in AUC-PR. Therefore,

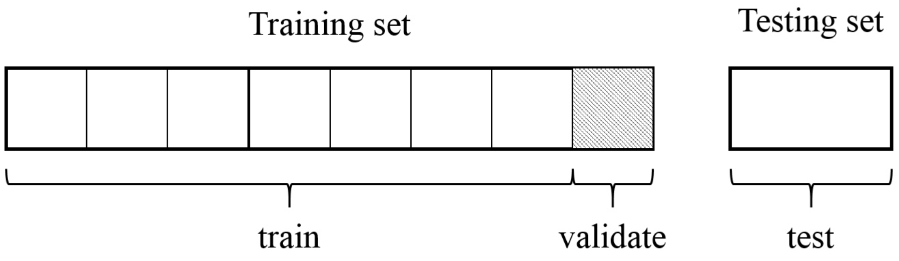


Figure 3: Example of the division of the dataset for cross-validation and testing

we chose to use AUC-PR as the last measure of comparison between the performance of the different hypothesis in cross-validation.

5.2. Cross-validation

From the three chosen families of models (Logistic Regression, Support Vector Machines and Random Forests), we want to know which can make the best predictions. Moreover, we want to determine the best parameters for each model.

The whole data set was previously split into training and testing sets. Now a further split is performed on the training set. Part is used in actual training, while the rest, which we call the validation set, is used to help us finding the best parameters of each model. We can estimate the performance by repeating the training-validation procedure multiple times. The most commonly applied validation technique is called *K-fold cross-validation*, in which the training set is divided in K folds, with each of the folds being left out at a time for the training-validation sequence, as illustrated in Figure 3.

Algorithm 1 describes the steps implemented to do cross-validation. This procedure was repeated for the three models: Random Forests, Support Vector Machines and Logistic Regression. The choice of the parameters to test was done by grid search, a technique which consists of performing an exhaustive search through a predefined space of parameters. In some cases, a refined grid search was additionally performed after finding a suitable subset of the parameter space.

Algorithm 1: Cross-validation

```
Data: training dataset
Result: Estimate of performance of models with different parameters
initialization;
foreach Parameter set do
    Load classifier with new parameters
    Split into K-fold validation sets
    for $k \in K$ do
        Train on train set
        Test on validation set
        Calculate performance measures - AUC-ROC and AUC-PR
    end
    Calculate mean and variance of the performance measures in the K validation runs
end
```

We have also compared results when training on a balanced sample (i.e. equal number of fraudulent and legitimate records) achieved by randomly under-sampling the legitimate records vs. an unbalanced sample of all records. Each training set had 347,572 records of which 312,814 were used for training and 34,758 for validation at a time. In the case of under-sampling, the records used for training were reduced to a number around 13,000 (circa two times the number of fraudulent cases in nine tenths of the records), while the validation was done on all 34,758 records from the validation fold. Random Forests and Logistic Regression were validated with both balanced and unbalanced data sets. Support Vector Machines has a much higher computational complexity and therefore it was just trained on the balanced sets generated by under-sampling.

Random Forests. Random Forests is an ensemble method proposed by [27] which consists of creating many decision trees and combining their estimates. The parameters which were varied were the minimum number of features to split each node of the tree (*Min. Split*), the criterion for quality of node split (ginni impurity or entropy), the number of trees, and whether a balanced set was used (*under-sampling*). The number of trees is not a true parameter, as more trees will always lead to a higher performance, but also more computational time. Nevertheless, with 1500 trees, the performance was practically stagnating, so this was the maximum value we tested. The top five results of this grid search can be seen on Table 1. Under-sampling did not lead to a better performance. In fact, there was no indication that using a balanced sample by under-sampling would achieve a different performance. Regarding the criterion for split, it is clear that the best performance is achieved by using the entropy criterion.

Support Vector Machines. Support Vector Machines is an advanced statistical classifier, which can make use of a kernel trick to map the data to a high dimensional feature space. We used an implementation of Support Vector Machines which yields a continuous probabilistic output. We employed a radial basis function kernel (*RBF*). The kernel coefficient Gamma and regularization term C were varied in a logarithmic scale. A high regularization term represents a weaker regularization. The best results were obtained for a value of C=10, as can be seen from Table 1.

Logistic Regression. Logistic Regression is a widely used method for classification and regression. Due to memory limitations, we could not train a logistic regression on a higher order representation of the features (e.g. second or third order transformation). Like in Support Vector Machines, we varied the regularization term. The only solver used was an implementation of the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (*lbfgs*) optimization algorithm.

Validation results. From Table 1, we can conclude that the performance of Support Vector Machines and Logistic Regression is similar. The best performance is achieved by Random Forests. The AUC-ROC is slightly higher than in the other two models and the AUC-PR is considerably higher. Therefore, this was the model selected.

5.3. Testing results

When testing, we want to estimate the performance in evaluating new data of the model which we chose in training – Random Forests – with the parameters which achieved the best performance. We finally come to use the *testing set* which we previously separated from the rest of our data. The performance in cross-validation has an optimistic bias, thus we expected to get a lower performance in testing.

Table 2 shows the results of testing. The complete curves can be seen on Figure 4 (Figure 4a represents the ROC curve, while Figure 4b exhibits the PR curve). Contrary to the performance results in validation (see Section 5.2), the value of the area under the precision-recall curve (AUC-PR) was not so satisfactory. However, Random Forests still presented a high value for the area under the receiver operating characteristic curve (AUC-ROC). Table 3 shows the results of the suspicion scores estimated by Random Forests on the testing data, divided by bins of 5 basis points. The ratio of frauds increases monotonously with the score, which is a sign that higher scores represent a higher probability of fraud.

The distribution of scores for legitimate and for fraudulent orders can be seen on the histograms in Figure 5. On the left, we see the distribution of the scores which were estimated for legitimate orders. On the right the scores estimated for fraudulent orders. The first thing to notice is that the distributions are different, which is a sign that the classifier recognized the two classes. There is a high occurrence of low scores for legitimate orders, which is what we would expect. On the other hand, fraudulent orders have a rather even distribution of scores. It would be expected that more fraudulent orders had high scores (near the value 1). This is likely to be a consequence of the unbalanced data set used for training, as the classifier can very well identify legitimate orders but has more difficulties with fraudulent observations.

In order to calculate business metrics such as the number of chargebacks (false negatives) and the number of refusals of legitimate payments (false positives), we must define a score threshold for considering a transaction legitimate or fraudulent based on its suspicion score. We detail two different approaches to defining this threshold, the

Table 1: Results of cross-validation

Panel A: Grid search results for Random Forests

Random Forests					
Num trees	Min. Split	Criterion	Under-sampling	AUC-PR	AUC-ROC
1500	10	entropy	No	0,479	0,935
1500	12	entropy	No	0,477	0,935
1500	11	entropy	No	0,475	0,935
1500	6	entropy	No	0,475	0,935
1500	16	entropy	No	0,474	0,935

Panel B: Grid search results for Support Vector Machines

Support Vector Machines					
C	Gamma	Kernel	Under-sampling	AUC-PR	AUC-ROC
10	0,0464	RBF	Yes	0,337	0,906
10	0,0215	RBF	Yes	0,336	0,902
10	0,01	RBF	Yes	0,319	0,896
10	0,1	RBF	Yes	0,317	0,903
1	0,1	RBF	Yes	0,305	0,895

Panel C: Grid search results for Logistic Regression

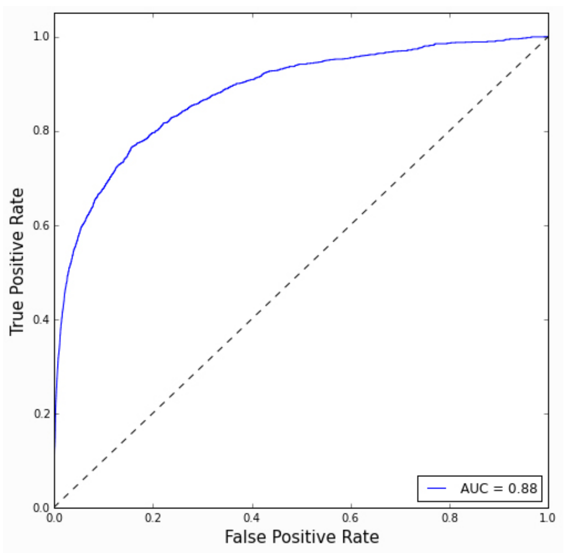
Logistic Regression					
C	Solver		Under-sampling	AUC-PR	AUC-ROC
100,00	lbfgs		No	0,360	0,907
3,16	lbfgs		No	0,357	0,905
1,00	lbfgs		No	0,349	0,902
0,32	lbfgs		No	0,324	0,895
3,16	lbfgs		Yes	0,313	0,903

Table 2: Testing results of the Random Forests model.

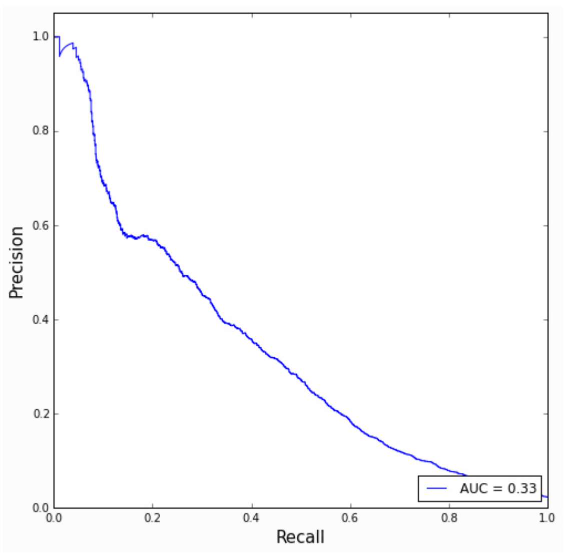
Classifier	AUC-PR	AUC-ROC
Random Forests	0,333	0,880

Table 3: Number of legitimate and fraudulent orders whose score falls into each range

Range	Total	Fraud	OK	Ratio
0 - 5	52469	180	52289	0,34
5 - 10	13510	159	13351	1,18
10 - 15	6810	127	6683	1,86
15 - 20	3828	115	3713	3
20 - 25	2941	111	2830	3,77
25 - 30	2197	109	2088	4,96
30 - 35	1445	106	1339	7,34
35 - 40	1022	112	910	10,96
40 - 45	914	161	753	17,61
45 - 50	564	129	435	22,87
50 - 55	365	110	255	30,14
55 - 60	213	86	127	40,38
60 - 65	194	102	92	52,58
65 - 70	190	85	105	44,74
70 - 75	111	57	54	51,35
75 - 80	59	51	8	86,44
80 - 85	43	42	1	97,67
85 - 90	17	17	0	100
90 - 95	1	1	0	100
95 - 100	0	0	0	-

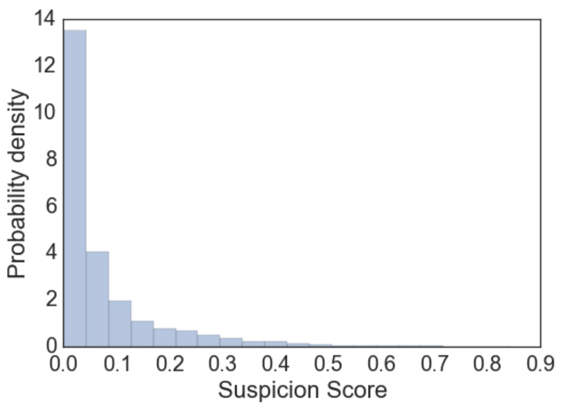


(a) ROC curve.

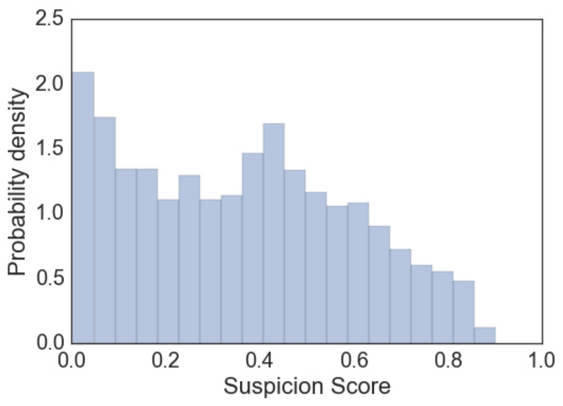


(b) Precision-Recall curve.

Figure 4: ROC and PR curves of testing results with Random Forests algorithm.



(a) Scores of legitimate orders.



(b) Scores of fraudulent orders.

Figure 5: Histograms of score predictions for each of the labels.

Table 4: Confusion matrix of results when setting threshold at c=0.22.

	Legitimate (actual)	Fraud (actual)	Total
Legitimate (predict)	77129 (88.76%)	618 (0.71%)	77747
Fraud (predict)	7904 (9.09%)	1242 (1.42%)	9146

first approach poses a scenario where manual revision is not possible and all orders are automatically approved and rejected. The second approach, explores the possibility of combining automatic classification with manual revision.

5.4. *Defining a score threshold – Approach 1*

This approach estimates the performance in the case that all orders are automatically approved or rejected in accordance with their suspicion score. One approach to define the threshold is to look for the value which originates the point of the curve closest to the top left corner in the ROC space. This is the point where Recall ($\frac{TruePositives}{TruePositives+FalseNegatives}$) and Specificity ($\frac{TrueNegatives}{TrueNegatives+FalsePositives}$) have the same value, which achieves the compromise between the two measures. Alternatively, different weights could be attributed to Recall or Specificity to find another threshold. Setting the threshold at this value, we can draw the confusion matrix of results which can be seen on Table 4.

In case this classifier would be processing all orders, the estimated number of chargebacks would be of 0.71% (number of false negatives), while 1.42% of all orders would be correctly rejected as fraudulent. However, we would incur in a high number of false positives (9.09%). This would lead to more than 10% of all orders being rejected. This is not acceptable in practice, and thus manual revision is still necessary for the orders considered fraudulent by the classifier. This approach is explored in Subsection 5.5.

5.5. *Defining a score threshold – Approach 2*

Our second approach consists in bringing together automatic classification with manual revision of orders. Hence, all orders with a score below a specific threshold are automatically approved, while the rest of the orders are manually revised. The specific threshold can be chosen by evaluating the trade-off between the cost of manually revising an order and the financial risk of fraud. At our case study company there were already resources available to revise most orders manually. In accordance with the company's objectives, we set the goal that only 20% of orders would be manually revised. Hence, the threshold was set at the value which is higher than 80% of the score of all orders. The advantage of this approach is that it can be easily adapted to new business requirements. If the company is interested in taking a lower risk, it can choose to decrease the number of automatically approved orders, incurring in higher manual revision costs. Figure 6 depicts this trade-off by showing how many fraudulent orders will be automatically approved if we vary the level of automation.

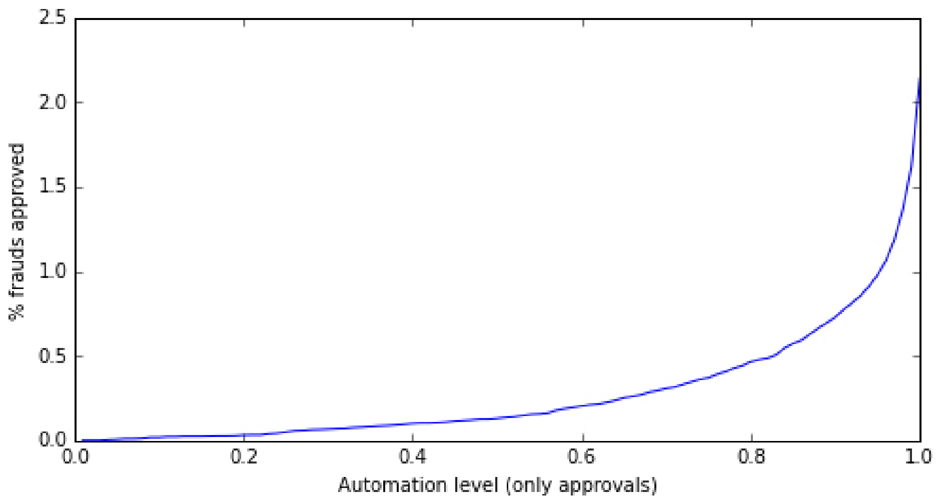


Figure 6: Plot of automation level and fraud.

Table 5: Confusion matrix of results when automatically approving 80% records with lowest estimated suspicion scores.

	Legitimate (actual)	Fraud (actual)	Total
Legitimate (predict)	83441 (96.02%)	768 (0.88%)	83914
Fraud (predict)	1592 (1.83%)	1092 (1.25%)	2684

Automation level:	80%
Recall:	0.587
Specificity:	0.981
Fallout:	0.019
Precision:	0.407

In order to build a confusion matrix we must estimate the effectiveness of manual revision. The following assumptions were made, based on past performance of manual revision at this company:

- The score threshold splits the orders which would be automatically approved (score under the threshold) from the ones which would be revised (score above the threshold);
- When a fraudulent order is manually revised, there is a 75% probability that it will be refused;
- When a legitimate order is manually revised, there is a 90% probability that it will be accepted.

Table 5 shows the results of this approach. The results show a high value of Specificity, which can be interpreted as the fraction of legitimate orders which are approved, in this case circa 98%. On the other hand, the Recall value is rather low: only circa 59% of the fraudulent orders would be refused. Precision is the fraction of fraudulent orders out of all which were refused ($\frac{TruePositives}{TruePositives+FalsePositives}$). Only 41% of the refused orders were actually fraudulent. Fallout can be interpreted as the "false alarm" rate, i.e. the probability of falsely rejecting a legitimate order ($\frac{FalsePositives}{TrueNegatives+FalsePositives}$). This value is equal to 1 minus the value of specificity, in this case circa 2%. This combination of the Random Forests classifier with the manual revision of orders, seems to yield very good results in increasing the automation level and having a low rate of customer insults (fallout). These values meet all the goals of the company.

Table 6: Relative importance of the top 10 attributes.

Importance	Feature	Description
13,3%	TimeSinceFirstOrder	Time elapsed since user's first purchase at this merchant
10,3%	OrderValue	Value of order
7,6%	Brand	All the features describing the product brand
6,3%	sCity_RiskLevel	Engineered variable - risk level associated with shipping city
6,1%	Quantity	Quantity of items in the order
5,9%	Gender	Gender of items in the order
5,4%	Similarity (UserName/CardName)	Engineered variable - similarity between username and name on credit card
4,9%	NumCardsUsed	Number of cards used by customer at this merchant
4,9%	PaymentAttempts	Payment attempts by user for this order
4,6%	OrderTimeGMT	Absolute time at which the order is placed

5.6. Variables importance

The Random Forests models allows us to estimate the relative importance of each of the features. We can see the importance of the top 10 attributes on Table 6. Attributes which were represented by many features such as the product brand have been aggregated into one feature in order to have a better overview of their importance. The importance of each variable is calculated as the "mean decrease impurity", described in [28].

We can see that the time since the customer's first order is the most important attribute. The value and quantity of items in the order also appear to be relevant features. On the product perspective, its brand and gender are together responsible for the same level of importance as the customer's first order date. The risk associated with the city where the order will be shipped to is the fourth most important attribute. A few features related to the payment itself also show significant importance. These include the similarity between the name on the credit-card and the name of the customer, the number of cards used by the customer in all his orders at this merchant, and the number of repeated payments attempts for this order.

Other variables such as the currency used or the similarity between billing and shipping address ranked lower in terms of relative importance. The information we get from this analysis can be of use to improve the manual revision process and in building new models in the future.

6. Deployment

The deployment of the selected model involves additional issues, such as the way the model will be updated and the integration with retailer's existing systems and services. This section explores these issues and shows some of the first results that the company is getting from using our solution.

6.1. Implementation and operation

In order to maintain the relevance of the classifier, it is important to update the data used for training. We created a recurring job on the database server to update the table where the data set is saved. The final model is to be integrated in the order classification service. This application will evaluate new orders and return a suspicion score. It contains two main procedures:

1. Training the algorithm
2. Classifying orders

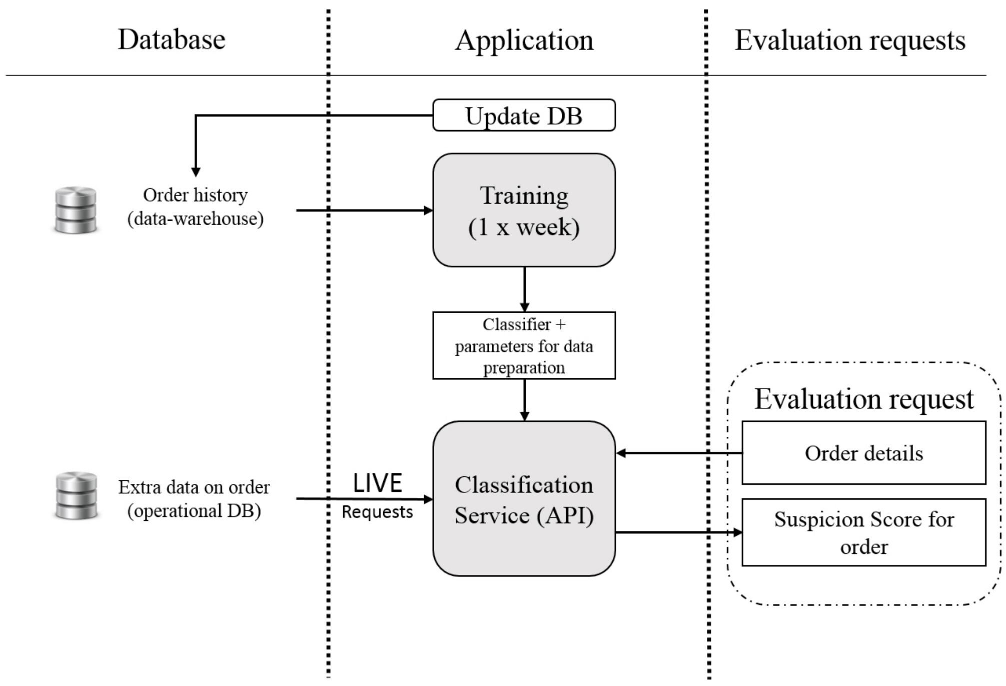

Database
Application
Evaluation requests
Update DB
Order history (data-warehouse)
Training (1 x week)
Classifier + parameters for data preparation
Evaluation request
Order details
Extra data on order (operational DB)
LIVE Requests
Classification Service (API)
Suspicion Score for order

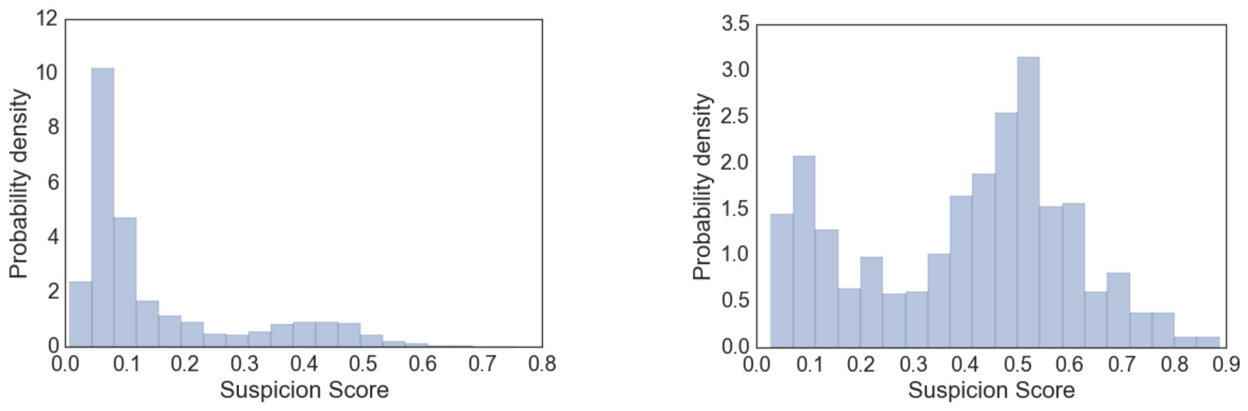
Figure 7: Diagram of the Order Classification System

Figure 7 illustrates the main parts of the system. Training the algorithm happens once every week. After the data set in the database has been updated, the training script will download the records. As it takes several weeks before a chargeback is submitted, the training script will only use data from orders up to four months before the current time. As there is no need for a *testing set*, all downloaded data is used in training. The output of the training process is a set of parameters for data transformation and a classifier which will generate the suspicion score.

The classification service runs continuously, except when loading new parameters after the training process occurs. When a new order is placed on the company's web portal, it is submitted for evaluation through the classification service's application programming interface (API). At first, the service parses the information on the request to map each field to the respective variable. Then, it applies the transformation functions to the data, so that it generates the exact same features as in training. The preprocessed parameters saved from the training, such as the risk-level mapping of countries are used in this step. Finally, with all the features in the right format, the classifier is called to estimate the suspicion score for the order. This score is returned as the answer of the classification request.

6.2. First results of the new system

During our tests, batches of 50 requests were processed in circa 90s. This is more than enough for a prompt response. Moreover, as the evaluation of each order does not depend on the score of previous orders, it is possible to parallelise the computation for faster response times (which could be critical in the mid-term future). Figure 8 shows the first results of scores evaluated by this service in real time at the company. At this stage, the classification service is taking no direct decision to approve or reject orders, but estimating the suspicion score. We will not know for sure which orders were fraudulent before a few months have passed, but we can compare the scores estimated for orders which were cancelled by the Order Processing Team for suspicion of fraud, against those orders which have been accepted. Figure 8a shows the distribution of scores for all those orders which were accepted. On the right, Figure 8b shows this distribution of scores for the orders which were not accepted. The clear difference in the distribution of scores in the two cases indicates that the classifier has distinguished between these orders. A further analysis should compare the results of orders which originated chargebacks.



(a) Orders not cancelled.

(b) Orders cancelled for fraud suspicion.

Figure 8: Comparison of scores distributions of orders approved vs. cancelled by Order Processing Team.

7. Conclusions and future work

This work addressed the problem of fraud detection for retailers doing business online. The project had the goal of designing, developing and implementing a risk scoring system (using data mining techniques) at an e-tail merchant. This paper can help researchers and practitioners to design and implement data mining based systems, as it describes the complete development process and addresses practical implementation issues.

The choice of which variables to use was very important and could be done thanks to the empirical experience provided by the fraud analysts at the case study company. However, the opposite has also happened. Our exploratory analysis gave some insights to the fraud analysts for improving their manual revision process. In that sense, it was important to explore the database and understand what other variables could be used that were not obvious at first.

The most relevant features turned out to be those which were engineered out of one or multiple base variables. The time since the customer's first order, the similarity measures between names or the grouping of cities by risk are examples of such features. Future work comparing different transformation functions and providing guidance on which features to engineer would be very useful for this area. We recommend studying the possibility of grouping addresses, which would require matching addresses that are spelled differently.

Concerning the core part of the system, the classifier, we have confirmed that supervised learning methods are applicable to fraud detection in an e-tail merchant setting. All the three machine learning algorithms tested (Logistic Regression, Support Vector Machines and Random Forests) provided good results. Random Forests achieved the highest performance. This algorithm seems to be very adequate to be used for fraud detection, not only because of its good performance, but also due to the ease of implementation and fast computation time even with large datasets. We concluded that using a balanced set of observations (under-sampling legitimate records) was not significantly different in performance than using the much bigger unbalanced full set of records. We suggest as future work to explore whether oversampling of fraudulent records would lead to a different conclusion.

The testing results with the Random Forests algorithm showed that such a model would perform well enough to be of practical use. The advantages of a data mining approach are the possibility of automatically processing a large volume of orders, allowing e-tail businesses to expand sustainably. When using a continuous classifier such as Random Forests, it is critical to choose the score threshold for considering an order to be legitimate or fraudulent. Our approach is based on the assumption that the merchant will manually revise orders with a score above the threshold, instead of directly cancelling them. We provide the merchant the possibility to define the threshold by choosing the share of orders which should be automatically processed. We recommend the exploration of other ways to choose the score threshold, such as methods based on the business targets for fraud tolerance and customer insult rates. Such a cost-based performance measure could be used in training the algorithm in order to further steer the learning process towards the intended business outcomes.

The results obtained, under the assumption that the 80% of the orders in the testing set with the lowest scores would be automatically approved and the rest reviewed manually, showed a very high number of legitimate orders approved and a false alarm rate of only circa 2%. On the other hand, the probability of rejecting a fraudulent order was only of circa 59%. Still, the number of chargebacks would be at 0.88% which is well under the industry average of 1.6% for international orders in North America [1].

The contribution for the company was the implementation of this approach as a risk scoring application. The system developed extracts the data from the database, prepares it, trains the algorithm and runs a web application. The latter gets requests with order details, and estimates the risk score for the order. At the end of this project, the system was implemented and the first results of real-time order evaluations are now being assessed.

Finally, this work details the many steps that need to be taken to develop a complete application of fraud detection. We hope that our case provided some insights into fraud detection in the perspective of merchants, and that researchers and practitioners will be motivated to further explore the problem of presenting solutions for fraud detection in e-tail.

Acknowledgments

The authors are thankful to the company staff, namely in the Operations and Business Intelligence departments, for the opportunity to collaborate on a management problem that is so interesting, and for their high commitment to the project. This was the primary reason why this project was a success. A word of appreciation goes also to José Borges and Carlos Soares, from INESC TEC and FEUP, for their various comments which helped to elaborate this study.

References

1. CyberSource, . 2013 online fraud report. Tech. Rep.; CyberSource; 2013. URL: `http://www.cybersource.com`.
2. Bhatla, T.P., Prabhu, V., Dua, A.. Understanding Credit Card Frauds. *Cards business review* 2003;1(6):1–15.
3. Montague, D.. Essentials of Online payment Security and Fraud Prevention. Essentials Series; Wiley; 2010. ISBN 9780470915141. URL: `https://books.google.pt/books?id=3IJCmhWztBIC`.
4. Quah, J.T., Sriganesh, M.. Real-time credit card fraud detection using computational intelligence. *Expert Systems with Applications* 2008;35(4):1721–1732. URL: `http://linkinghub.elsevier.com/retrieve/pii/S0957417407003995`. doi:`10.1016/j.eswa.2007.08.093`.
5. Ghosh, , Reilly, . Credit card fraud detection with a neural-network. *1994 Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences* 1994;3:621–630. doi:`10.1109/HICSS.1994.323314`.
6. Fawcett, T., Provost, F.. Adaptive Fraud Detection. *Data Mining and Knowledge Discovery* 1997;316(1):291–316. doi:`10.1023/A:1009700419189`.
7. Srivastava, A., Kundu, A., Sural, S., Member, S.. Credit Card Fraud Detection Using Hidden Markov Model. *Ieee Transactions on Dependable and Secure Computing* 2008;5(1):37–48. URL: `http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4358713&url=http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4358713`. doi:`10.1109/TDSC.2007.70228`.
8. Whitrow, C., Hand, D.J., Juszczak, P., Weston, D., Adams, N.M.. Transaction aggregation as a strategy for credit card fraud detection. *Data Mining and Knowledge Discovery* 2009;18(1):30–55. URL: `http://www.springerlink.com/index/10.1007/s10618-008-0116-z`. doi:`10.1007/s10618-008-0116-z`.
9. Krivko, M.. A hybrid model for plastic card fraud detection systems. *Expert Systems with Applications* 2010;37(8):6070–6076. URL: `http://dx.doi.org/10.1016/j.eswa.2010.02.119`. doi:`10.1016/j.eswa.2010.02.119`.
10. Bhattacharyya, S., Jha, S., Tharakunnel, K., Westland, J.C.. Data mining for credit card fraud: A comparative study. *Decision Support Systems* 2011;50(3):602–613. URL: `http://linkinghub.elsevier.com/retrieve/pii/S0167923610001326`. doi:`10.1016/j.dss.2010.08.008`.
11. Jha, S., Guillen, M., Christopher Westland, J.. Employing transaction aggregation strategy to detect credit card fraud. *Expert Systems with Applications* 2012;39(16):12650–12657. URL: `http://dx.doi.org/10.1016/j.eswa.2012.05.018`. doi:`10.1016/j.eswa.2012.05.018`.
12. Brabazon, A., Cahill, J., Keenan, P., Walsh, D.. Identifying online credit card fraud using artificial immune systems 2010;:1–7doi:`10.1109/CEC.2010.5586154`.
13. Phua, C., Lee, V., Smith, K., Gayler, R.. A comprehensive survey of data mining-based accounting-fraud detection research. *International Conference on Intelligent Computation Technology and Automation* 2010;1:50–53. doi:`10.1109/ICICTA.2010.831`.
14. Ngai, E., Hu, Y., Wong, Y., Chen, Y., Sun, X.. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems* 2011;50(3):559–569. URL: `http://linkinghub.elsevier.com/retrieve/pii/S0167923610001302`. doi:`10.1016/j.dss.2010.08.006`.
15. Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R.. CRISP-DM 1.0 step-by-step data mining guide. Tech. Rep.; The CRISP-DM consortium; 2000. URL: `http://www.crisp-dm.org/CRISPWP-0800.pdf`.
16. Cortes, C., Pregibon, D., Volinsky, C.. Communities of interest. Springer; 2001.
17. Moreau, Y., Lerouge, E., Verrelst, H., Stormann, C., Burge, P., Leuven, K.U.. A hybrid system for fraud detection in mobile communications. *Neural Networks* 1999;(April):447–454.
18. Chan, P.K., Stolfo, S.J.. Toward Scalable Learning with Non-uniform Class and Cost Distributions : A Case Study in Credit Card Fraud Detection 1 Introduction. *n Proceedings of the Fourth In- ternational Conference on Knowledge Discovery and Data Mining* 1998;:164–168.

19. Meyer, D., Leisch, F., Hornik, K.. The support vector machine under test. *Neurocomputing* 2003;55(12):169 – 186. URL: `http://www.sciencedirect.com/science/article/pii/S0925231203004314`. doi:`http://dx.doi.org/10.1016/S0925-2312(03)00431-4`; support Vector Machines.
20. Rossum, G.. Python reference manual. Tech. Rep.; Amsterdam, The Netherlands, The Netherlands; 1995.
21. McKinney, W.. Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference* 2010;1697900(Scipy):51–56. URL: `http://conference.scipy.org/proceedings/scipy2010/mckinney.html`.
22. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 2011;12:2825–2830.
23. Miner, G., Nisbet, R., Elder IV, J.. Handbook of statistical analysis and data mining applications. Academic Press; 2009.
24. Fawcett, T.. An introduction to ROC analysis. *Pattern Recognition Letters* 2006;27(8):861–874. doi:`10.1016/j.patrec.2005.10.010`.
25. Hanley, J.A., McNeil, B.J.. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology* 1982;143(1):29–36.
26. Davis, J., Goadrich, M.. The Relationship Between Precision-Recall and ROC Curves. *Proceedings of the 23rd International Conference on Machine learning – ICML'06* 2006;:233–240URL: `http://portal.acm.org/citation.cfm?doid=1143844.1143874`. doi:`10.1145/1143844.1143874`.
27. Breiman, L.. Random forests. *Machine learning* 2001;:5–32URL: `http://link.springer.com/article/10.1023/A:1010933404324`. doi:`10.1023/A:1010933404324`. `arXiv:/dx.doi.org/10.1023%2FA%3A1010933404324`.
28. Breiman, L., Friedman, J., Stone, C., Olshen, R.. Classification and Regression Trees. The Wadsworth and Brooks-Cole statistics-probability series; Taylor & Francis; 1984. ISBN 9780412048418. URL: `https://books.google.pt/books?id=JwQx-WOmSyQC`.