Data Engineering for Fraud Detection

Bart Baesens[a,b], Sebastiaan Höppner[c], Tim Verdonck[d,c,*]

[a] *KU Leuven, Faculty of Economics and Business, Naamsestraat 69, Leuven 3000, Belgium*
[b] *University of Southampton, School of Management, Highfield Southampton, SO17 1BJ, United Kingdom*
[c] *KU Leuven, Department of Mathematics, Celestijnenlaan 200B, Leuven 3001, Belgium*
[d] *University of Antwerp, Department of Mathematics, Middelheimlaan 1, Antwerp 2020, Belgium*

Abstract

Financial institutions increasingly rely upon data-driven methods for developing fraud detection systems, which are able to automatically detect and block fraudulent transactions. From a machine learning perspective, the task of detecting suspicious transactions is a binary classification problem and therefore many techniques can be applied. Interpretability is however of utmost importance for the management to have confidence in the model and for designing fraud prevention strategies. Moreover, models that enable the fraud experts to understand the underlying reasons why a case is flagged as suspicious will greatly facilitate their job of investigating the suspicious transactions. Therefore, we propose several data engineering techniques to improve the performance of an analytical model while retaining the interpretability property. Our data engineering process is decomposed into several feature and instance engineering steps. We illustrate the improvement in performance of these data engineering steps for popular analytical models on a real payment transactions data set.

Keywords: Decision analysis, Payment transactions fraud, Instance engineering, Feature engineering, Cost-based model evaluation.

1. Introduction

The association of certified fraud examiners (ACFE) estimates that a typical organization loses 5% of its revenues to fraud each year. The fifth oversight report on card fraud analyses developments in fraud related to card payment schemes (CPSs) in the Single Euro Payments Area (SEPA), issued in September 2018 by the European Central Bank and covering almost the entire card market, indicates that the total value of fraudulent transactions conducted using cards issued within SEPA and acquired worldwide amounted to 1.8 billion Euros in 2016, which in relative terms, i.e. as

*Corresponding author: Tim Verdonck

Email addresses: bart.baesens@kuleuven.be (Bart Baesens), sebastiaan.hoppner@kuleuven.be (Sebastiaan Höppner), Tim.Verdonck@uantwerpen.be (Tim Verdonck)

a share of the total value of transactions, amounted to 0.041% in 2016 (European Central Bank, September 2018). These are just a few numbers to indicate the severity of the payment transactions fraud problem. It is also seen that losses due to fraudulent activities keep increasing each year and affect card holders worldwide. Therefore, fraud detection and prevention are more important than ever before and developing powerful fraud detection systems is of crucial importance to many organizations and firms in order to reduce losses by timely blocking, containing and preventing fraudulent transactions.

The Oxford Dictionary defines fraud as follows: *the crime of cheating somebody in order to get money or goods illegally.* This definition captures the essence of fraud and covers the many different forms and types of fraud. On the other hand, it does not very precisely describe the nature and characteristics of fraud and as such does not provide much direction for discussing the requirements of a fraud detection system. A more thorough and detailed characterization of the multifaceted phenomenon of fraud is provided by Van Vlasselaer et al. (2017): *Fraud is an uncommon, well-considered, imperceptibly concealed, time-evolving and often carefully organized crime which appears in many types of forms.* This definition highlights five characteristics that are associated with particular challenges related to developing a fraud detection system.

The first emphasized characteristic and associated challenge concerns the fact that fraud is uncommon. Independent of the exact setting or application, only a small minority of the involved population of cases typically concerns fraud, of which furthermore only a limited number will be known to be fraudulent. This makes it difficult to both detect fraud, since the fraudulent cases are covered by the non-fraudulent ones, as well as to learn from historical cases to build a powerful fraud detection system since only few examples are available. This will make it hard for machine learning techniques to extract meaningful patterns from the data.

Fraud is also imperceptibly concealed since fraudsters exactly try to blend into their environments to remain unnoticed. This relates to the subtlety of fraud since fraudsters try to imitate normal behavior. Moreover, fraud is well-considered and intentional and complex fraud structures are carefully planned upfront. Fraudsters can also adapt or refine their tactics whenever needed, for example, due to changing fraud detection mechanisms. Therefore, fraud detection systems need to improve and learn by example.

The traditional approach to fraud detection is expert-driven, which builds on the experience, intuition, and business or domain knowledge of one or more fraud investigators. Such expert-based rule base or engine is typically hard to build and maintain. A shift is occurring toward data-

driven or machine learning based fraud detection methodologies. This shift is triggered by the digitization of almost every aspect of society and daily life, which leads to an abundance of available data. Financial institutions increasingly rely upon data-driven methods for developing powerful fraud detection systems, which are able to automatically detect and block fraudulent transactions. In other words, we need adaptive analytical models to complement experience-based approaches for fighting fraud. A stream of literature has reported upon the adoption of data-driven aproaches for developing fraud detection systems (Phua et al., 2010; Ngai et al., 2011). These methods significantly improve the efficiency of fraud detection systems and are easier to maintain and more objective. From a machine learning perspective, the task of detecting fraudulent transactions is a binary classification problem.

A natural first step to move from expert-based approaches to data driven techniques (while still taking into account the experience of the fraud experts) is to consider logistic regression and/or decision trees. These simple analytical models can then be replaced by complex techniques such as random forests and boosting methods, support vector machines, neural networks and deep learning to increase the detection power. Although the latter are definitely powerful analytical techniques, they suffer a very important drawback which is not desirable from a fraud prevention perspective: they are black box models which means that they are very complex to interpret. We would also like to note that these complex models not always significantly outperform simple analytical models such as logistic regression (Baesens et al., 2003; Lessmann et al., 2015) and we strongly believe that you should always start with implementing these simple techniques. Many benchmarking studies have illustrated that complex analytical techniques only provide marginal performance gains on structured, tabular data sets as frequently encountered in common classification tasks such as fraud detection, credit scoring and marketing analytics (Baesens et al., 2003; Lessmann et al., 2015). It is our firm belief that in order to improve the performance of any analytical model, we should focus more on the data itself rather than developing new, complex predictive analytical techniques. This is exactly the aim of data engineering. It can be defined as the clever engineering of data hereby exploiting the bias of the analytical technique to our benefit, both in terms of accuracy and interpretability at the same time. Often times it will be applied in combination with simple analytical techniques such as linear or logistic regression so as to maintain the interpretability property which is so often needed in analytical modeling. In our context of fraud analytics, interpretability is of key importance to design smart fraud prevention mechanisms. Data engineering can be decomposed into feature engineering and instance engineering. Feature engineering aims at designing smart features in one of two possible

ways: [either by transforming existing features using smart transformations or by designing better, new features which will allow a simple analytical technique such as linear or logistic regression to boost its performance, or by designing new features (a process often called featurization) to basically achieve the same aim.] Instance engineering entails the careful selection of instances or observations again with the aim to improve predictive modeling performance. Put differently, it aims at selecting those observations which positively contribute to the learning of the analytical technique and remove those that have a detrimental impact on it. Obviously, this is not a trivial exercise and many instance engineering techniques have been developed which we will carefully study and experiment with in this paper. In this paper the focus will be on successful data engineering steps to improve the performance of a fraud detection model. More concretely, we will describe the lessons that we have learnt when complementing expert-based approaches with machine learning or data-driven techniques to combat payment transactions fraud for a large European bank.

This paper is organized as follows. We start with presenting our data engineering process: Section 2 presents feature engineering steps whereas instance engineering is explained in Section 3. In Section 4 popular performance measures in an (imbalanced) classification setting are described. In Section 5, more information about payment transaction fraud and the observed data set is given. This section also illustrates the benefits of the various data engineering steps by showing increased performance on our real data set. Finally, concluding remarks and potential directions for future research are provided in Section 6.

2. Feature engineering

The main objective of machine learning is to extract patterns to turn data into knowledge. Since the beginning of this century, technological advances have drastically changed the size of data sets as well as the speed with which these data must be analyzed. Modern data sets may have a huge number of instances, a very large number of features, or both. In most applications, data sets are compiled by combining data from different sources and databases (containing both structured and unstructured data) where each source of information has its strengths and weaknesses. Before applying any machine learning algorithm, it is therefore necessary to transform these raw data sources into interesting features that better help the predictive models. This essential step, which is often denoted feature engineering, is of utmost importance in the machine learning process. We believe that data scientists should be well aware of the power of feature engineering and that they should share good practices.

An important set of interesting features can be created based on the famous Recency, Frequency, Monetary (RFM) principle. Recency measures how long ago a certain event took place, whereas frequency counts the number specific events per unit of time. Besides recency features, we also present several other time-related features. Features related to monetary value measure the intensity of a transaction, typically expressed in a currency such as Euros or USD. We also introduce features based on unsupervised anomaly detection and briefly discuss some other advanced feature engineering techniques.

2.1. Frequency features

We explain the idea behind the RFM principle by first deriving frequency features using a transaction aggregation strategy in order to capture a customer's spending behavior. This methodology was first proposed by Whitrow et al. (2009) and has been used by a number of studies (Bhattacharyya et al., 2011; Jha et al., 2012; Dal Pozzolo et al., 2014; Bahnsen et al., 2016). Frequency calculates how many transactions were made during a sliding time window that satisfies predefined conditions, as illustrated in Figure 1. The first step in creating frequency features consists in aggregating the transactions made during the last given time period (e.g. last 3 months), first by card or account number, then by payment channel, authentication method, beneficiary country or other, followed by counting the number of transactions. It is important to choose an appropriate time period over which to aggregate a customer's transactions. When time passes, the spending patterns of a customer are not expected to remain constant over the years. For transactions made with debit cards, we propose to use a fixed time frame of 90, 120 or 180 days ($\sim$ 3, 4 or 6 months). Let $\mathcal{D}$ denote a set of N transactions where each transaction is represented by the pair $(\boldsymbol{x}_i, y_i)$ for $i = 1, 2, \ldots, N$. Here $y_i \in \{0, 1\}$ describes the true class of transfer i and $\boldsymbol{x}_i = \left(x_i^1, x_i^2, \ldots, x_i^p\right)$ represents the p associated features of transfer i. Bahnsen et al. (2016) describe the process of creating frequency features as selecting those transactions that were made in the previous t_p days, for each transaction i in the data

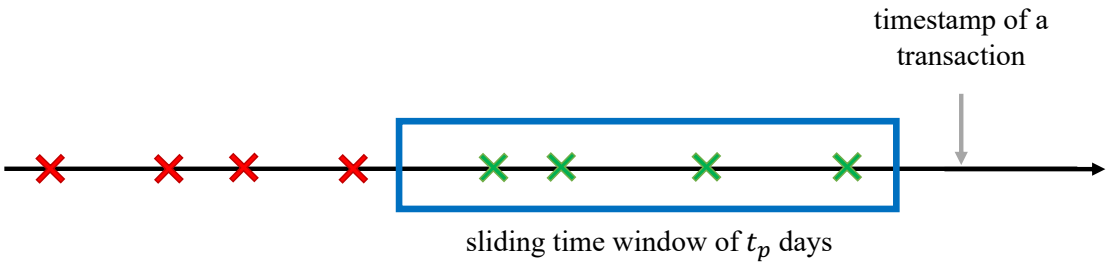

Figure 1: Timeline of transactions of a customer using, for example, a particular payment channel.

set $\mathcal{D}$,

$$\begin{aligned}\mathcal{D}_{t_p,i}^{freq} &= AGG^{freq}\left(\mathcal{D}, i, t_p\right) \\ &= \left\{x_j^{amt} \middle| \left(x_j^{id} = x_i^{id}\right) \text{ and } \left(days\left(x_i^{time}, x_j^{time}\right) < t_p\right)\right\}_{j=1}^{N}\end{aligned} \tag{1}$$

where $AGG(\cdot)$ is a function that aggregates transactions of $\mathcal{D}$ into a subset associated with a transaction i with respect to the time frame t_p; x_i^{time} is the timestamp of transaction i; x_i^{amt} is the amount of transaction i; x_i^{id} is the customer or card identification number of transaction i; and $days(t_1, t_2)$ is a function that calculates the number of days between the times t_1 and t_2. Finally, the frequency feature is calculated as

$$x_i^{freq} = \left|\mathcal{D}_{t_p,i}^{freq}\right| \tag{2}$$

where $|\cdot|$ is the cardinality of a set. This aggregation strategy, however, does not take the combination of different features into account. For example, we can aggregate transactions according to certain criteria, such as: transactions made in the last t_p days using the same authentication method (e.g. pin code or fingerprint) and the same payment channel (e.g. online banking or mobile app). For calculating such features, Bahnsen et al. (2016) expand (1) as follows

$$\begin{aligned}\mathcal{D}_{t_p,i}^{freq2} =& AGG^{freq}\left(\mathcal{D}, i, t_p, cond1, cond2\right) \\ =& \left\{x_j^{amt} \middle| \left(x_j^{id} = x_i^{id}\right) \text{ and } \left(days\left(x_i^{time}, x_j^{time}\right) < t_p\right)\right. \\ & \left. \text{and } \left(x_j^{cond1} = x_i^{cond1}\right) \text{ and } \left(x_j^{cond2} = x_i^{cond2}\right)\right\}_{j=1}^{N}\end{aligned} \tag{3}$$

where $cond1$ and $cond2$ could be one of the features of a transaction (e.g. authentication method, payment channel, beneficiary country, etc.). Similarly, the frequency feature is then calculated as

$$x_i^{freq2} = \left|\mathcal{D}_{t_p,i}^{freq2}\right|. \tag{4}$$

One could also define new features as the ratio of frequency features. For example,

$$x_i^{ratio} = x_i^{freq2} / x_i^{freq} \tag{5}$$

which is always between 0 and 1. Since x_i^{ratio} is the fraction of of transfers for which conditions $cond1$ and $cond2$ hold over all transactions in the past t_p days, this feature represents the probability that both conditions $cond1$ and $cond2$ are met by the customer.

We show an example to further clarify how the frequency features are calculated. Consider a set of transactions made by a customer between 01/07/2019 and 03/07/2019, as shown in Table 1. Then we estimate the frequency features x_i^{freq} and x_i^{freq2} by setting $t_p = 1$ day ($\sim$ 24 hours) for ease of calculation.

The frequency features give us specific details about the spending behavior of the customer. For example, if a customer frequently used a particular payment channel in the past t_p days, its frequency is obviously large. However, a zero frequency for a particular payment channel implies that the customer has not used that payment channel in the past t_p days which indicates anomalous behavior and perhaps fraud. The total number of frequency features can grow quite quickly, as t_p can have several values, and the combination of criteria can be quite large as well. For the experiments we set the different values of t_p to 90, 120 and 180 days. Then we calculate the frequency features using (2) and (4) as well as (5) with the aggregation criteria including payment channel, authentication method, beneficiary country, type of communication, and others.

Initial features						Frequency features	
TransId	CustId	Timestamp		Authentication method	Payment channel	x_i^{freq}	x_i^{freq2}
1	1	01/07/2019	16:51	pin code	web	0	0
2	1	01/07/2019	19:04	pin code	web	1	1
3	1	01/07/2019	19:36	fingerprint	app	2	0
4	1	01/07/2019	23:31	pin code	web	3	2
5	1	02/07/2019	17:48	fingerprint	app	3	1
6	1	02/07/2019	22:12	fingerprint	app	2	1
7	1	02/07/2019	23:34	fingerprint	app	2	2
8	1	03/07/2019	01:40	pin code	app	3	0

Table 1: Example calculation of frequency features: x_i^{freq} is the number of transactions in the last 24 hours, and x_i^{freq2} is the number of transactions with the same authentication method and payment channel in the last 24 hours.

2.2. Recency features

Although frequency features are powerful in describing a customer's spending behavior, they do not take the aspect of time into account. Recency features are a way to capture this information. Recency measures the time passed since the previous transaction that satisfy predefined conditions. To explain how recency features are defined we show an example where we create a recency feature derived from the authentication method used by the customer as illustrated in Figure 2. When a customer makes a transfer $\boldsymbol{x}_i$, she chooses a method x_i^{AU} to authenticate herself. Examples of authentication methods are passwords, pin codes, fingerprints, itsme[1], iris scans and hardware tokens.

[1]This is a popular app in Belgium that allows you to safely, easily and reliably confirm your (digital) identity and approve transactions.

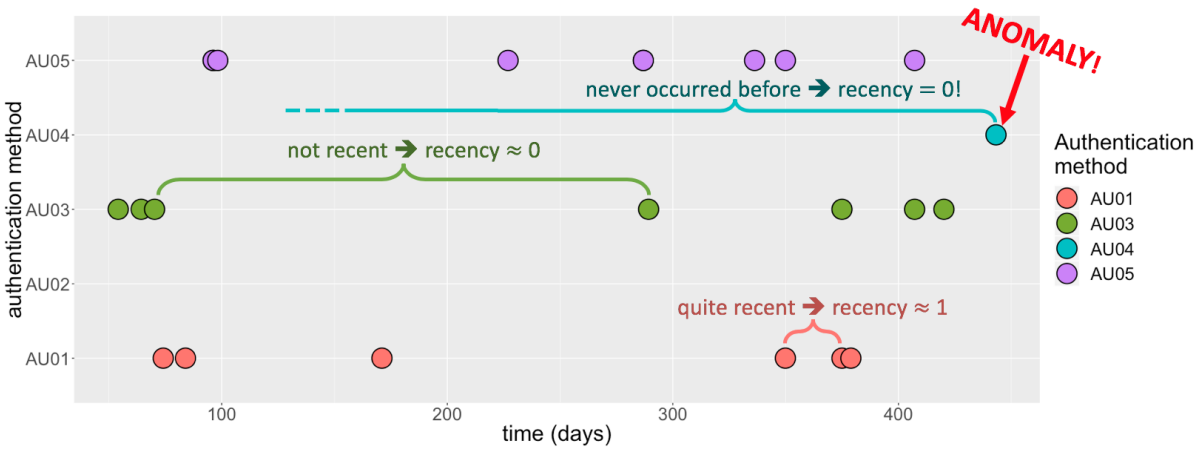

Figure 2: Example of a recency feature derived from the authentication method used by a customer. When the customer makes a transaction, she chooses one of five possible authentication methods which are labeled as AU01, AU02, ..., AU05. If the time between the same two successive authentication methods is long, the recency is close to zero, while if that time is short, the recency is close to 1. If an authentication method is used for the first time, its recency is defined as zero.

For each transaction i in the data set $\mathcal{D}$, we define the recency of the transaction's authentication method as

$$x_i^{AU,recency} = \exp\left(-\gamma \cdot \Delta t_i\right) \text{ where} \tag{6}$$

$$\Delta t_i = \min\left\{ days\left(x_i^{time}, x_j^{time}\right) \mid \left(x_j^{id} = x_i^{id}\right) \text{ and } \left(x_j^{AU} = x_i^{AU}\right) \right\}_{j=1}^{N}.$$

Here Δt_i is the time interval, typically in days, between two consecutive transfers made by the same customer with identification number x_i^{id} using the same authentication method x_i^{AU}. The parameter γ can be chosen such that, for example, the recency is small (e.g. 0.01) when $\Delta t = 180$ days ($\sim$ 6 months) in which case $\gamma = -\log(0.01)/180 = 0.026$. Notice that recency is always a number between 0 and 1. When the time period Δt between two consecutive transfers with the same authentication method is small (large), we say that the authentication method has (not) recently been used. In that case the recency for this authentication method is close to one (zero). When an authentication method is used for the first time, we define its recency to be zero. A zero or small recency shows atypical behavior and might indicate fraud. Figure 3 shows that recency indeed decreases when the time interval becomes larger. The parameter γ determines how fast the recency decreases. For larger values of γ, recency will decrease quicker with time and vice versa.

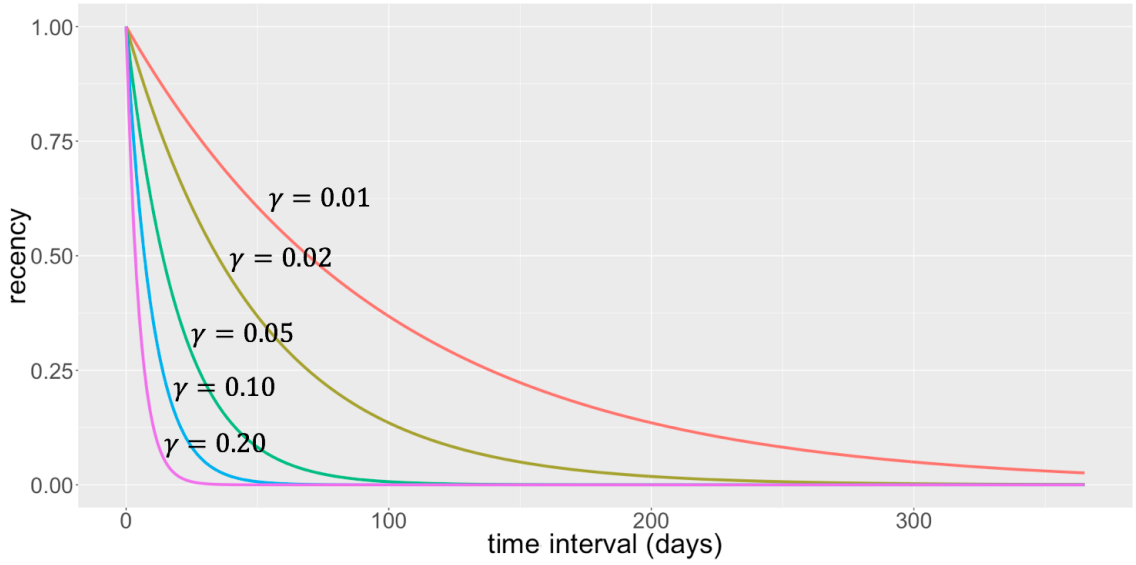

Figure 3: Recency versus time (in days) for different values of γ.

2.3. Other time-related features

It is well-known that time is an important aspect in fraud detection. Besides recency features other time-related features can be created based on the assumption that certain events, like a customer who makes transactions, occur at similar moments in time. Having a transaction at 22:00 might be very regular for one person, but very suspicious for another person. Since, for every customer, we know the timestamps of all their transactions in the past, we can use this information to decide whether a new transaction at 22:00 is atypical for a particular customer. For the set of timestamps of transactions made by a each customer we can construct a circular histogram, as shown in Figure 4 (left). Since 00:00 is the same as 24:00, we have to model the time of a transaction as a periodic variable by fitting an appropriate statistical distribution (Bahnsen et al., 2016). A popular choice is the von Mises distribution, also known as the periodic normal distribution because it represents a normal distribution wrapped around a circle (Fisher, 1995). The von Mises distribution of a set of timestamps $\mathcal{D}^{time} = \{t_1, t_2, \ldots, t_N\}$ is defined as

$$\mathcal{D}^{time} \sim von\ Mises(\mu, \kappa) \tag{7}$$

where parameters μ and $1/\kappa$ represent the periodic mean and the periodic standard deviation, respectively. These parameters can easily be estimated by most statistical software. We use the function `mle.vonmises` from the R package `circular` to compute the maximum likelihood estimates for the parameters of a von Mises distribution.

For each customer we construct a confidence interval for the time of a transaction. First, we select the set of transactions made by the same customer in the last t_p days,

$$\begin{aligned}\mathcal{D}^{time}_{t_p,i} &= AGG^{time}\left(\mathcal{D}, i, t_p\right) \\ &= \left\{x^{time}_j \middle| \left(x^{id}_j = x^{id}_i\right) \text{ and } \left(days\left(x^{time}_i, x^{time}_j\right) < t_p\right)\right\}_{j=1}^{N}.\end{aligned} \tag{8}$$

Based on this set of selected timestamps, the estimated parameters $\hat{\mu}$ and $\hat{\kappa}$ are calculated. Next, a von Mises distribution is fitted on the set of timestamps using these estimates:

$$x^{time}_i \sim von\ Mises\left(\hat{\mu}\left(\mathcal{D}^{time}_{t_p,i}\right), \hat{\kappa}\left(\mathcal{D}^{time}_{t_p,i}\right)\right). \tag{9}$$

Once the von Mises distribution is fitted on the timestamps of the customer's transactions we can construct a confidence interval with probability α, e.g. 80%, 90%, 95%. An example is presented in Figure 4 (right). Using the confidence interval, a binary feature is created: a transaction is flagged as normal or suspicious depending on whether or not the time of the transaction is within the confidence interval. Table 2 shows an example of a binary feature that takes the value of one if the current time of the transaction is within the confidence interval of the time of the previous transactions with a confidence of $\alpha = 0.9$. Of course, multiple of these binary features can be extracted for different values of α and time period t_p. The new feature also helps to get a better understanding of when a customer is expected to make transactions. Note that this feature (just as many others) solely indicates atypical behavior for a customer, which might give an indication for fraud. If a certain

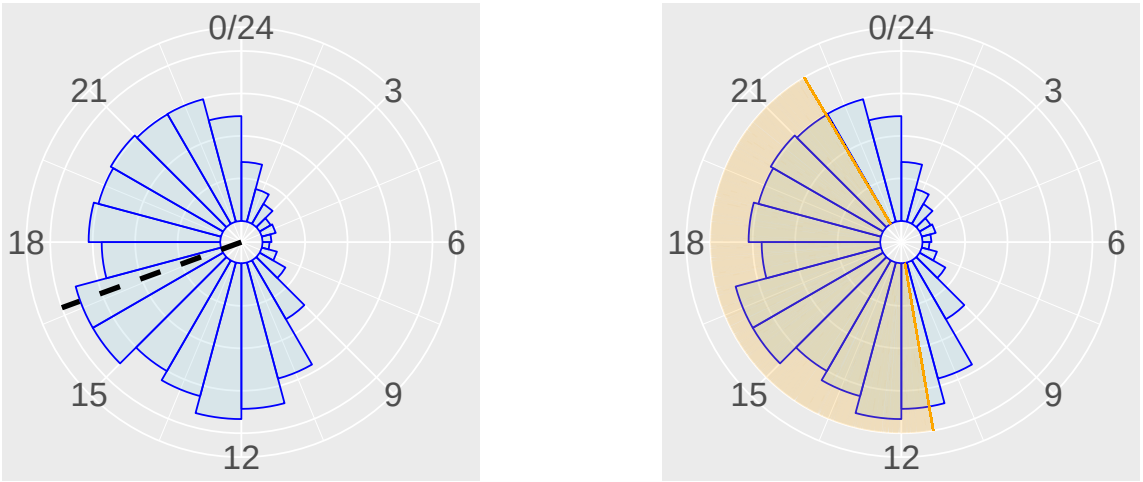

Figure 4: (Left) Circular histogram of timestamps of transactions. The dashed line is the estimated periodic mean of the von Mises distribution. (Right) Circular histogram including the 90% confidence interval (orange area).

transaction is flagged as potentially fraudulent due to this feature, then it is important that this information is also given to the fraud investigators. If they see that the customer is abroad, then that could be the reason for the atypical value of this feature.

Instead of looking at the timestamp of a transaction within a day, we can of course create similar features indicating how atypical it is for a customer to have a payment on a certain day or above a certain amount. Some customers, for example, may only do transactions during the weekend. Adding such features based on customer spending history may bring significant increase in model performance. Most predictive models let you also easily evaluate which features increased the performance of your model and which are not significant for discriminating frauds from non-frauds.

TransId	Time	Periodic mean	Confidence interval	Binary feature
1	01/07/2019 16:51	-	-	-
2	01/07/2019 19:04	-	-	-
3	01/07/2019 19:36	17:57	16:07 - 19:48	1
4	01/07/2019 23:31	18:31	16:32 - 20:29	0
5	02/07/2019 17:48	19:40	15:39 - 23:40	1
6	02/07/2019 22:12	19:14	15:27 - 23:01	1
7	02/07/2019 23:34	19:47	15:52 - 23:42	1
8	03/07/2019 01:40	20:21	16:05 - 00:38	0

Table 2: Example calculation of a binary feature that informs whenever a transaction is being made within the confidence interval (with $\alpha = 0.9$) of the time of the previous transactions.

2.4. Monetary value related features

The last pillar of the RFM principle involves monetary value related features which focus on the amount that is transferred. Monetary features calculate various statistics such as the total value, the average, and the standard deviation of the transferred amounts that were pursued during the

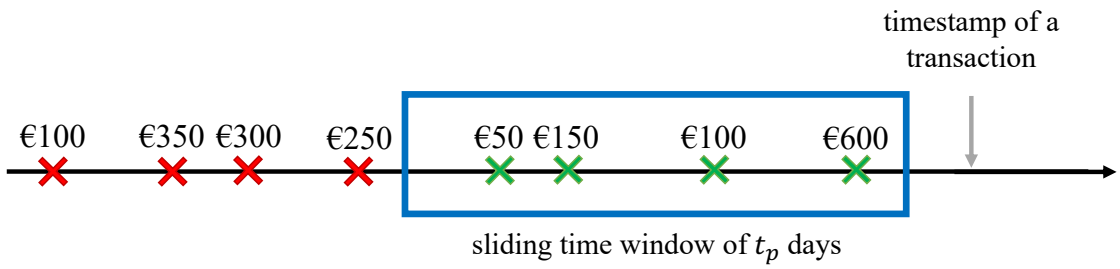

Figure 5: Timeline of amounts transferred by a customer using, for example, a particular payment channel.

sliding time window that satisfy predefined conditions (Figure 5). The first step in creating monetary features is the same as with frequency features: select those transactions that were made in the last t_p days, as in (1). Next, we can calculate the total amount spent on those transactions,

$$x_i^{total} = \sum_{j=1}^{N} x_j^{amt} I\left(x_j^{amt} \in \mathcal{D}_{t_p,i}^{freq}\right) \tag{10}$$

where $I(\cdot)$ is the indicator function. Of course, we can also aggregate transactions according to certain criteria, as in (3), followed by calculating their sum,

$$x_i^{total2} = \sum_{j=1}^{N} x_j^{amt} I\left(x_j^{amt} \in \mathcal{D}_{t_p,i}^{freq2}\right). \tag{11}$$

Transferring 500 Euros may be little for one person, but a lot for another person. A monetary feature that calculates the so-called *z-score* of an amount can indicate whether the amount is atypical for a particular customer. For a set of amounts $\mathcal{D}_{t_p,i}^{freq}$, the standardized values or z-scores are defined as

$$z_i = \frac{x_i^{amt} - \hat{\mu}_{\mathcal{D}}}{\hat{\sigma}_{\mathcal{D}}} \tag{12}$$

where $\hat{\mu}_{\mathcal{D}}$ and $\hat{\sigma}_{\mathcal{D}}$ are the sample mean and sample standard deviation, respectively,

$$\hat{\mu}_{\mathcal{D}} = Mean\left(\mathcal{D}_{t_p,i}^{freq}\right) \quad \text{and} \quad \hat{\sigma}_{\mathcal{D}} = Stdev\left(\mathcal{D}_{t_p,i}^{freq}\right). \tag{13}$$

As a rule of thumb, an amount is flagged as an outlier if its z-score is larger than 3, $|z_i| > 3$. Now consider the transactions made by a customer, as shown in Figure 6. The last amount of 500 Euros is clearly an outlier compared to the previous amounts. However, when using the sample mean and sample standard deviation, the z-score of the atypically high amount is only 2.66 and is therefore not regarded as abnormal.

Instead of computing the z-score using traditional estimates such as sample mean and sample standard deviation, we propose using robust alternatives such as the median and the median absolute deviation (MAD),

$$z_i^r = \frac{x_i^{amt} - \hat{\mu}_{\mathcal{D}}^r}{\hat{\sigma}_{\mathcal{D}}^r} \tag{14}$$

with

$$\hat{\mu}_{\mathcal{D}}^r = Median\left(\mathcal{D}_{t_p,i}^{freq}\right) \quad \text{and} \quad \hat{\sigma}_{\mathcal{D}}^r = MAD\left(\mathcal{D}_{t_p,i}^{freq}\right) \tag{15}$$

where

$$MAD\left(\{x_1, x_2, \ldots, x_n\}\right) = 1.4826 \cdot Median\left(\left|x_i - Median(\{x_j\}_{j=1}^n)\right|_{i=1}^n\right). \tag{16}$$

The constant scale factor 1.4826 ensures that the MAD is a consistent estimator for the estimation of the standard deviation σ, i.e. $\mathbb{E}\left[MAD\left(\{X_1, X_2, \ldots, X_n\}\right)\right] = \sigma$ for X_j distributed as $N\left(\mu, \sigma^2\right)$ and large n. Using the robust estimates, the z-score of the last amount in Figure 6 is 5.79, which clearly indicates that the 500 Euros is atypical for this customer.

Remark: transferred amounts are often right-skewed as shown in Figure 7 (left). The rule of thumb, i.e. $|z_i| > 3$, implicitly assumes that the z-scores are distributed as $N\left(\mu, \sigma^2\right)$. Before standardizing the amounts, a transformation is often applied to them that changes their distribution to one that resembles a normal distribution, or at least a symmetric distribution. One such transformation is the natural logarithm, as shown in Figure 7 (right).

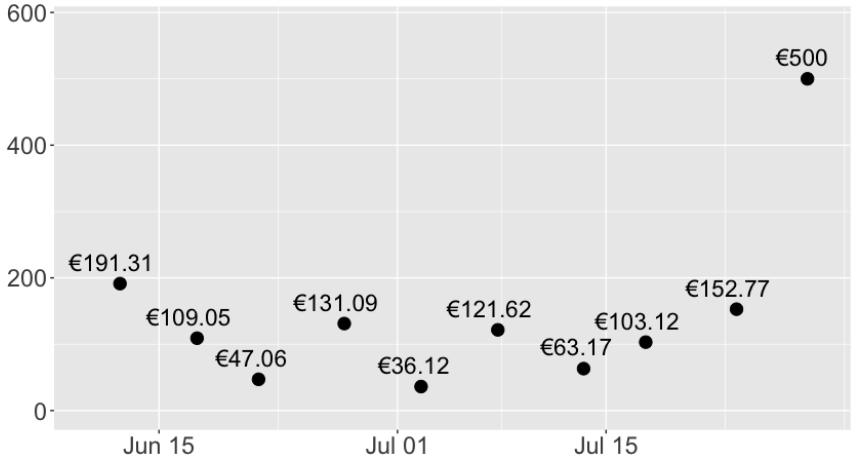

Figure 6: An example of transferred amounts. The last amount of 500 Euros is clearly an outlier compared to the previous amounts. The atypical high amount is not indicated when using traditional estimates such as sample mean and sample standard deviation. Instead, we have to use robust estimates such as the median and the median absolute deviation (MAD).

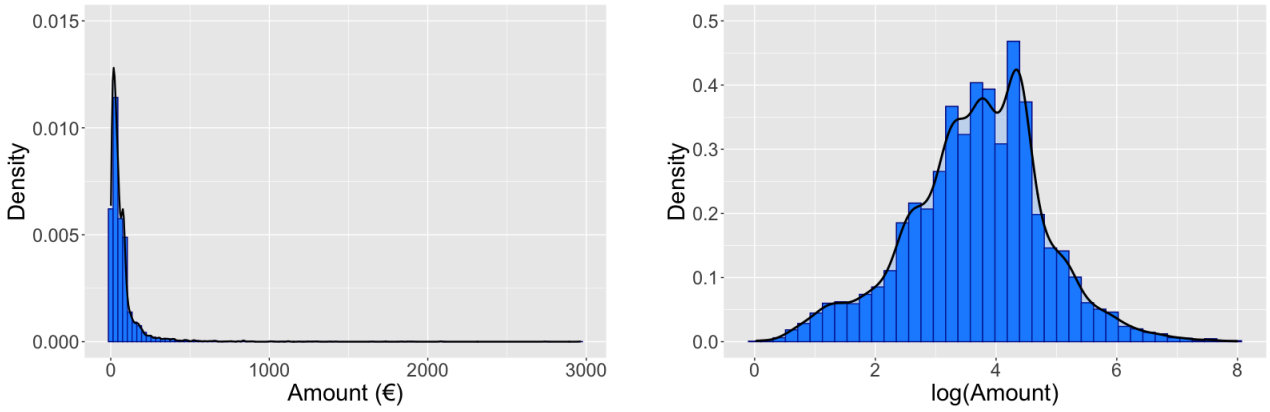

Figure 7: Histogram and kernel density estimate of amounts (left) and natural logarithm of those amounts (right).

A popular alternative for computing (robust) z-scores is the boxplot, which is a very popular graphical tool to analyze a univariate data set (Tukey, 1977). The boxplot marks all observations outside the interval $[Q_1 1.5\text{IQR}; Q_3 + 1.5\text{IQR}]$ as potential outliers, where Q_1, Q_2 and Q_3 denote respectively the first, second (or median) and third quartile and $\text{IQR} = Q_3 - Q_1$ equals the interquartile range. It is known that the boxplot typically flags too many points as outlying when the data are skewed and therefore Hubert and Vandervieren (2008) have modified the boxplot interval so that the skewness is sufficiently taken into account.

In practice one often tries to detect outliers using diagnostics starting from a classical or traditional fitting method. Unfortunately, these traditional techniques can be affected by outliers so strongly that the resulting fitted model may not allow to detect the deviating observations. This is called the masking effect (see e.g. Rousseeuw and Leroy (2005)). Additionally, some good data points might even appear to be outliers, which is known as swamping (Davies and Gather, 1993). To avoid these effects, the goal of robust statistics is to find a fit which is close to the fit we would have found without the outliers. We can then automatically identify the outliers by their large 'deviation' (e.g., their distance or residual) from that robust fit. It is not our aim to replace traditional techniques by a robust alternative, but we have illustrated that robust methods can give you extra insights in the data and may improve the reliability and accuracy of your analysis.

2.5. Features based on (unsupervised) anomaly detection techniques

In this section we focus on unsupervised techniques that do not use the target variable (fraudulent or not). Anomaly detection techniques flag anomalies or outliers, which are observations that deviate from the pattern of the majority of the data. These flagged observations indicate atypical behavior and hence may contain crucial information for fraud detection and should be investigated by the fraud expert. As an alternative, we propose to use the *outlyingness score or metric* of several anomaly detection techniques as features that we add to our data set.

Anomalies in a single dimension (i.e. univariate outliers) can be detected by computing (robust) z-scores (and see which observations are in absolute value larger than 3) or by constructing the (adjusted) boxplot (and see which observations are outside the boxplot interval or fence). Another tool for univariate anomaly detection that is also popular in fraud detection is Newcomb-Benford law, which makes predictions about the distribution of the first leading digit of all numbers (Nigrini, 2012; Barabesi et al., 2018). These techniques can then be applied on each feature in the data set. However, in this way it is only possible to detect anomalies that are atypical in (at least) one dimension or feature of our data set. Since fraudsters succeed very well in blending in with legitimate customers,

they are typically not detected by checking each feature separately. It is important to flag those observations that deviate in several dimensions from the main data structure but are not atypical in one of the features. Such multivariate outliers can only be detected in the multidimensional space and require the use of advanced models.

A first tool for this purpose is robust statistics, which first fits the majority of the data and then flags the observations that deviate from this robust fit (Rousseeuw and Hubert, 2018). For a multivariate $n \times p$ data set $\boldsymbol{X}$, one can calculate the robust Mahalanobis distance (or robust generalized distance) for each observation $\boldsymbol{x}_i$:

$$MD(\boldsymbol{x}_i, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}) = \sqrt{(\boldsymbol{x} - \hat{\boldsymbol{\mu}})^T \hat{\boldsymbol{\Sigma}}^{-1} (\boldsymbol{x} - \hat{\boldsymbol{\mu}})}. \tag{17}$$

An observation is then flagged as anomaly if its distance exceeds the cut-off value $\sqrt{\chi^2_{p,0.975}}$, which is the 0.975 quantile of the chi-squared distribution with p degrees of freedom. It is of utmost importance that robust estimates of multivariate location and scatter are used in the computation of the distances (to avoid masking and swamping effects). A popular method yielding such estimates is the Mininimum Covariance Determinant (MCD) method of Rousseeuw and Driessen (1999) or the Minimum Regularized Covariance Determinant (MRCD) estimator of Boudt et al. (2020) in case of high-dimensional data. Note that also various robust alternatives for popular predictive models are proposed in literature. These robust supervised techniques automatically flag anomalies (typically with a convenient graphical tool to visualize the anomalies). Therefore it is interesting to also apply robust versions of the predictive models on the data and carefully examine the anomalies flagged with these techniques (for more information see e.g. Maronna et al. (2019); Heritier et al. (2009); Atkinson and Riani (2000)). Recently, Rousseeuw et al. (2019) also used robust statistics to detect potential fraud cases in time series of imports into the European Union.

Besides robust statistics, many other unsupervised anomaly detection tools from various research fields have been proposed (Goldstein and Uchida, 2016). We briefly introduce and illustrate three popular techniques: k-nearest neighbors distance (Angiulli and Pizzuti, 2002; Brito et al., 1997), local outlier factor (LOF) (Breunig et al., 2000) and isolation forests (Liu et al., 2008). The k-nearest neighbors distance for an observation is the average distance to each of its k closest neighbors. This distance measures how isolated an observation is from its neighbors and hence a large distance typically indicates an anomaly. The LOF score is the average density around the k nearest neighbors divided by the density around the observation itself and anomalies typically have a score above one. Isolation forest is obtained by taking an ensemble of isolation trees which try to isolate each observation as quickly as possible. The final score is the average of the standardized path length (i.e.

number of splits to isolate the observation) over all trees. Hence for all the methods above it holds: the higher the score or metric, the more suspicious is the observation.

2.6. Other feature engineering techniques

In this paper, we only study a few feature engineering techniques to illustrate their importance as a key data engineering mechanism. Other powerful feature engineering techniques are the Box-Cox and Yeo-Johnson transformation which both univariately transform data variables so as to boost the performance of the predictive analytical model. Note that these transformation techniques are sensitive to outliers and will try to move outliers inward at the expense of the normality of the central part of the data. Therefore various robust transformation procedures have been proposed in literature (see e.g. Carroll and Ruppert (1985); Riani (2008); Marazzi et al. (2009); Raymaekers and Rousseeuw (2020)). Feature engineering techniques have also been designed for unstructured data such as text, network data, and multimedia data (e.g., images, audio, videos). For text data, one commonly uses Singular Value Decomposition (SVD) or Natural Language Processing (NLP) as feature engineering techniques. For network data, node2vec and GraphSage (Grover and Leskovec, 2016; Hamilton et al., 2017) have proven to be very valuable techniques. Deep learning has been used to learn complex features for multimedia data. As an example, convolutional neural networks can learn key features to describe objects in images. However, an important caveat is that many of these features are black box in nature and thus hard to interpret for business decision makers. Finally, tailored feature engineering techniques have been designed for specific domains, e.g., Item2Vec in Recommender Systems (Barkan and Koenigstein, 2016).

3. Instance engineering

A major challenge in fraud analytics is the imbalance or skewness of the data, meaning that typically there are plenty of historical examples of non-fraudulent cases, but only a limited number of fraudulent cases. For example, in a credit card fraud setting, typically less than 0.5% of transactions are fraudulent. Such a problem is commonly referred to as the needle in a haystack problem, and might cause an analytical technique to experience difficulties in learning to create an accurate model. Every classifier faced with a skewed data set typically tends to favor the majority class. In other words, the classifier tends to label all transactions as non-fraudulent since it then already achieves a classification accuracy of more than 99%. Classifiers typically learn better from a more balanced distribution. Two popular ways to accomplish this is by undersampling, whereby non-fraudulent

transactions in the training set are removed, or oversampling, whereby fraudulent transactions in the training set are replicated.

A practical question concerns the optimal, non-fraud/fraud odds, which should be the goal by doing under- or oversampling. This of course depends on the data characteristics and quality and type of classifier. Although train and error is commonly adopted to determine this optimal odds, the ratio 90% non-fraudsters versus 10% fraudsters is usually already sufficient for most business applications.

The Synthetic Minority Oversampling technique, or SMOTE, is another interesting approach to deal with skewed class distributions (Chawla et al., 2002). In SMOTE, the minority class is oversampled by adding synthetic observations. The creation of these artifical fraudsters goes as follows. In Step 1 of SMOTE, for each minority class observation, the k nearest neighbors (of same class) are determined. Step 2 then randomly selects one of the neighbors and generates synthetic observations as follows: 1) take the difference between the features of the current minority sample and those of its nearest neighbor. 2) multiply this difference with a random number between 0 and 1 and 3) add the obtained result as new observation to the sample, hereby increasing the frequency of the minority class.

The key idea of these undersampling and oversampling techniques is to adjust the class priors to enable the analytical technique to create a meaningful model that discriminates the fraudsters from the non-fraudsters. By doing so, the class posteriors become biased. This is not a problem if the fraud analyst is interested in ranking the observations in terms of their fraud risks. However, if well-calibrated fraud probabilities are needed, then the posterior probabilities can be adjusted (Saerens et al., 2002).

Since its introduction in 2002, many variants of SMOTE have been proposed in literature (see e.g. Zhu et al. (2019) and Kovács (2019) for an overview). In this paper, we visually show the differences between ADASYN (He et al., 2008), MWMOTE (Barua et al., 2012) and ROSE (Lunardon et al., 2014) and show their performance on our data set. We refer to their papers for details. It is clear that there is not one oversampling technique that always yield the best result (Amin et al., 2016).

4. Measuring performance

The aim of detecting transfer fraud is to identify transactions with a high probability of being fraudulent. From the perspective of machine learning, the task of predicting the fraudulent nature of transactions can be presented as a binary classification problem where observations (i.e. transactions,

customers, etc.) belong either to class 0 or to class 1. We follow the convention that the fraudulent observations belong to class 1, whereas the legitimate observations correspond to class 0. We often speak of positive (class 1) and negative (class 0) observations.

Consider again our set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ of N transactions. In general, a classification algorithm provides a continuous score $s_i := s(\boldsymbol{x}_i) \in [0, 1]$ for each transaction i. This score s_i is a function of the observed features $\boldsymbol{x}_i$ of transaction i and represents the fraud propensity of that transaction. Here we assume that legitimate transfers (class 0) have a lower score than fraudulent ones (class 1). The score s_i is then converted to a predicted class $\hat{y}_i \in \{0, 1\}$ by comparing it with a classification threshold $t \in [0, 1]$. If a transfer's probability of being fraudulent as estimated by the classification model lies above this threshold value, then the transfer is predicted as fraud ($s_i > t \Rightarrow \hat{y}_i = 1$), and otherwise it is classified as legitimate ($s_i \leq t \Rightarrow \hat{y}_i = 0$).

A classification exercise typically leads to a confusion matrix as shown in Table 3. Based on the confusion matrix, we can compute several performance measure such as *Precision*, *Recall* (also

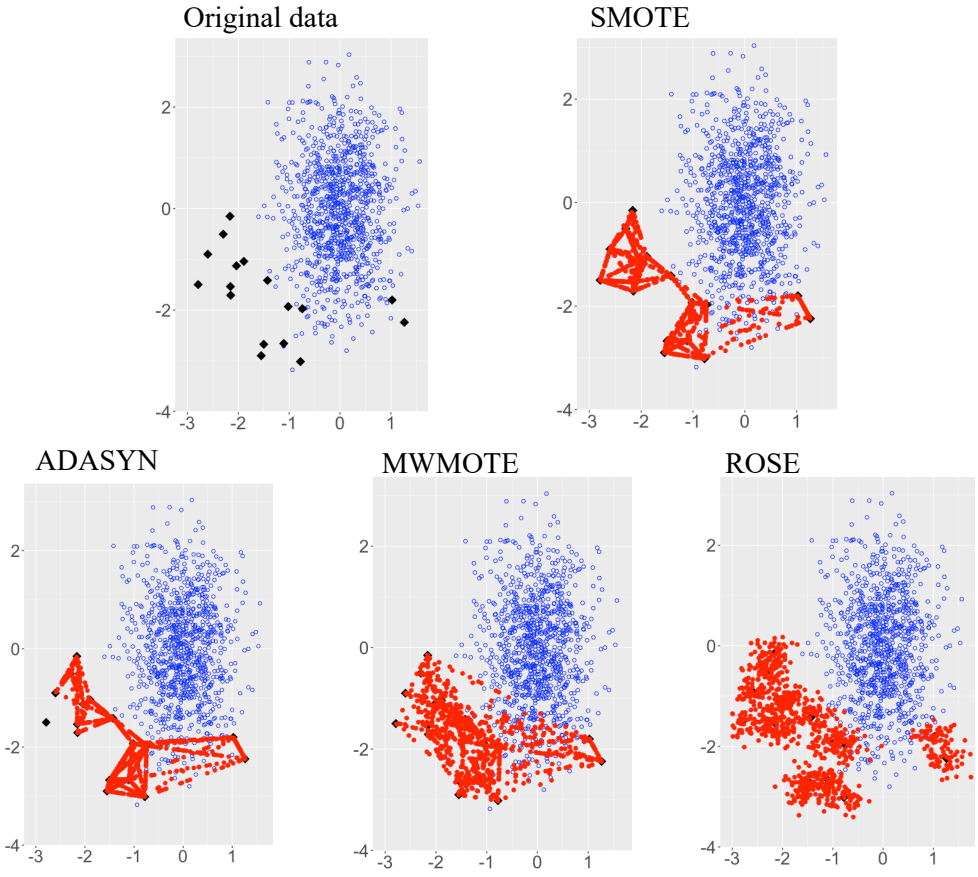

Figure 8: Illustration of SMOTE, ADASYN, MWMOTE and ROSE. The blue circles represent the legitimate cases, the black squares are the original fraud cases, and the red dots are the synthetic fraud cases.

	Actual legitimate (negative) $y = 0$	Actual fraudulent (positive) $y = 1$
Predicted as legitimate (negative) $\hat{y} = 0$	True negative (TN)	False negative (FN)
Predicted as fraudulent (positive) $\hat{y} = 1$	False positive (FP)	True positive (TP)

Table 3: Confusion matrix of a binary classification task.

called *True Positive Rate*, *Sensitivity* or *Hit Rate*), *False Positive Rate*, and F_1*-measure*. Each of these measures are calculated for a given confusion matrix that is based on a certain threshold value $t \in [0, 1]$.

The receiver operating characteristic (ROC) curve, as shown on the left plot in Figure 9, is probably the most popular method to analyze the effectiveness of a classifier. The ROC curve is obtained by plotting for each possible threshold value the false positive rate (FPR) on the X-axis and the true positive rate (TPR) on the Y-axis. As a graphical tool the ROC curve visualizes the tradeoff between achieving a high recall (TPR) while maintaining a low false positive rate (FPR), and is often used to find an appropriate decision threshold. Provost et al. (1998) argue that ROC curves, as an alternative to accuracy estimation for comparing classifiers, would enable stronger and more general conclusions. For more information about ROC curves we refer to Krzanowski and Hand (2009) and Swets (2014).

Comparing classifiers based solely on their ROC curves can be challenging. Therefore, the ROC

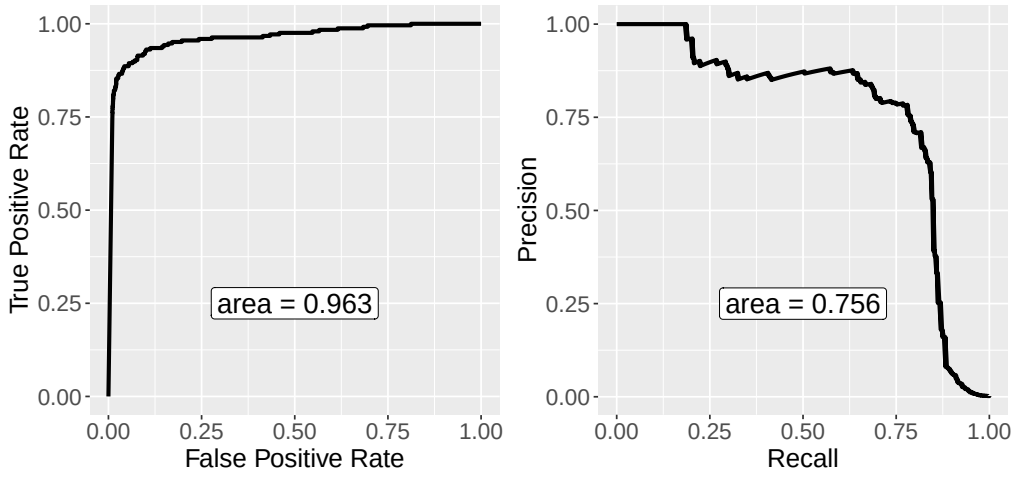

Figure 9: (Left) example of a ROC curve. (Right) example of a Precision-Recall curve. Both curves are based on the same classifier validated on the same data set.

curve is often summarized in a single score, namely the Area Under the ROC Curve (AUC) which varies between 0 and 1 (Fawcett, 2004, 2006; Ling et al., 2003). In the context of fraud detection, the AUC of a classifier can be interpreted as being the probability that a randomly chosen fraud case is predicted a higher score than a randomly chosen legitimate case. Therefore, a higher AUC indicates superior classification performance. A perfect classifier would achieve an AUC of 1 while a random model (i.e. no prediction power) would yield an AUC of 0.5.

When dealing with highly imbalanced data as is the case with fraud detection, AUC (and ROC curves) may be too optimistic and the Area under the Precision-Recall Curve (AUPRC) gives a more informative picture of a classifier's performance (Davis and Goadrich, 2006; Saito and Rehmsmeier, 2015; Fernández et al., 2018). As the name suggest, the Precision-Recall curve (right plot in Figure 9) plots the precision (Y-axis) against the recall (X-axis) or each possible threshold. The AUPRC is therefore also a value between 0 and 1. Both ROC and PR curves use the recall, but the ROC curve also plots the FPR whereas PR curves focus on precision. In the denominator of FPR, one sums the number of true negatives and false positives. In highly imbalanced data, the number of negatives (legitimate observations) is much larger than the number of positives (fraudulent observations) and hence the number of true negatives is typically very high compared to the number of false positives. Therefore, a large increase or decrease in the number of false positives will have almost no impact on FPR in the ROC curves. Precision, on the other hand, compares the number of true positives to the number of false positives and hence copes better with the imbalance between positive and negative observations. Since precision is more sensitive to class imbalance, the area under the Precision-Recall curve (AUPRC) is better to highlight differences between models for highly imbalanced data sets.

Despite the many ways to evaluate a classification model's performance we argue that the true business objective of a fraud detection system is to minimize the financial losses due to fraud. How-

	Actual legitimate (negative) $y_i = 0$	Actual fraudulent (positive) $y_i = 1$
Predicted as legitimate (negative) $\hat{y}_i = 0$	True negative $[C_i(0\|0) = 0]$	False negative $[C_i(0\|1) = A_i]$
Predicted as fraudulent (positive) $\hat{y}_i = 1$	False positive $[C_i(1\|0) = c_f]$	True positive $[C_i(1\|1) = c_f]$

Table 4: Cost matrix where, between square brackets, the related instance-dependent classification costs for transfer fraud are given.

ever, the performance measures mentioned so far do not incorporate any costs related to incorrect predictions such as not detecting a fraudulent transaction. Therefore, they may not be the most appropriate evaluation criteria when evaluating fraud detection models. In fact, the previous performance measures tacitly assume that misclassification errors carry the same cost, similarly with the correctly classified transactions. This assumption clearly does not hold in practice because wrongly predicting a fraudulent transaction as legitimate carries a significantly different financial cost than the inverse case. To better align the assessment of data-driven fraud detection systems with the actual objective of decreasing losses due to fraud, we extend the confusion matrix in Table 3 by incorporating costs as proposed in (Baesens et al., 2020). Let $C_i(\hat{y}|y)$ be the cost of predicting class $\hat{y}$ for a transfer i when the true class is y. If $\hat{y} = y$ then the prediction is correct, while if $\hat{y} \neq y$ the prediction is incorrect. In general, the costs can be different for each of the four cells in the confusion matrix and can even be instance-dependent, in other words, specific to each transaction i as indicated in Table 4. Hand et al. (2008) proposed a cost matrix, where in the case of a false positive (i.e. incorrectly predicting a transaction as fraudulent) the associated cost is the administrative cost $C_i(1|0) = c_f$. This fixed cost c_f has to do with investigating the transaction and contacting the card holder. When detecting a fraudulent transfer, the same cost $C_i(1|1)$ is allocated to a true positive, because in this situation, the card owner will still need to be contacted. In other words, the action undertaken by the company towards an individual transaction i comes at a fixed cost $c_f \geq 0$, regardless of the nature of the transaction. However, in the case of a false negative, in which a fraudulent transfer is not detected, the cost is defined as the amount $C_i(0|1) = A_i$ of the transaction i. The instance-dependent costs are summarized in Table 4. We argue that the cost matrix in Table 4 is a reasonable assumption. However, one could alter the cost matrix, for example, by using a variable cost for false positives that reflects the level of friction that the card holder experiences.

Using the instance-dependent cost matrix in Table 4, Bahnsen et al. (2016) define the cost of using a classifier $s(\cdot)$ on the transactions in $\mathcal{D}$ as

$$\begin{aligned} Cost\left(s\left(\mathcal{D}\right)\right) &= \sum_{i=1}^{N} \Bigg(y_i \Big[\hat{y}_i C_i(1|1) + (1-\hat{y}_i) C_i(0|1) \Big] \\ &\qquad + (1-y_i) \Big[\hat{y}_i C_i(1|0) + (1-\hat{y}_i) C_i(0|0) \Big] \Bigg) \\ &= \sum_{i=1}^{N} y_i (1-\hat{y}_i) A_i + \hat{y}_i c_f. \end{aligned} \tag{18}$$

In other words, the total cost is the sum of the amounts of the undetected fraudulent transactions ($y_i = 1$, $\hat{y}_i = 0$) plus the administrative cost incurred. The total cost may not always be easy to

interpret because there is no reference to which the cost is compared (Whitrow et al., 2009). So Bahnsen et al. (2016) proposed the *cost savings* of a classification algorithm as the cost of using the algorithm compared to using no algorithm at all. The cost of using no algorithm is

$$Cost_l(\mathcal{D}) = \min\{Cost(s_0(\mathcal{D})),\ Cost\,(s_1(\mathcal{D}))\} \tag{19}$$

where s_0 refers to a classifier that predicts all the transactions in $\mathcal{D}$ as belonging to class 0 (legitimate) and similarly s_1 refers to a classifier that predicts all the transfers in $\mathcal{D}$ as belonging to class 1 (fraud). The cost savings is then expressed as the cost improvement of using an algorithm as compared with $Cost_l\,(\mathcal{D})$,

$$Savings\,(s(\mathcal{D})) = \frac{Cost_l\,(\mathcal{D}) - Cost\,(s\,(\mathcal{D}))}{Cost_l\,(\mathcal{D})}. \tag{20}$$

In the case of transaction fraud, the cost of not using an algorithm is equal to the sum of amounts of the fraudulent transactions, $Cost_l\,(\mathcal{D}) = \sum_{i=1}^{N} y_i A_i$. The savings are then calculated as

$$Savings\,(s\,(\mathcal{D})) = \frac{\sum_{i=1}^{N} y_i \hat{y}_i A_i - \hat{y}_i c_f}{\sum_{i=1}^{N} y_i A_i}. \tag{21}$$

In other words, the costs that can be saved by using an algorithm are the sum of amounts of detected fraudulent transactions minus the administrative cost incurred in detecting them, divided by the sum of amounts of the fraudulent transactions.

Besides obtaining the best statistical accuracy or the highest cost savings, there are many other reasons why one model might be preferred above another, such as interpretability, operational efficiency and economical cost.

Interpretability refers to the intelligibility or readability of the analytical model. Models that enable the user to understand the underlying reasons why the model signals a case to be suspicious are called white-box models. Complex incomprehensible mathematical models are often referred to as black-box models. It might well be, in a fraud detection setting, that black-box models are acceptable, although in most settings, some level of understanding and in-fact validation, which is facilitated by interpretability, is required for the management to have confidence and allow the effective implementation of the model. In most situations, the aim of the fraud detection system is to select out of millions of payments the transactions that are most suspicious. These top, say 100, most suspicious transactions are then given to the fraud investigators for further examination. When using white box models, it is straightforward to also give information about why a certain transaction is flagged as being suspicious. This of course facilitates the job of the fraud investigators leading to more suspicious transactions that can be examined for example in one day. The need of

interpretability on the operator side, which advocates for relatively simple models and methods, has also the advantage to simplify for the end-user (a bank) the implementation, maintainability and possibility to update/enrich the system over time.

Operational efficiency refers to the response time or the time that is required to evaluate the model, or in other words, the time required to evaluate whether a case is suspicious. It also entails the efforts needed to collect and preprocess the data, evaluate the model, monitor and back-test the model, and re-estimate it when necessary. Operational efficiency can be a key requirement, meaning that the fraud detection system might have only a limited amount of time available to reach a decision and let a transaction pass or not. In others words, huge volumes of data need to be processed in a short time span. For example, in a credit card fraud detection setting, the decision time must typically be less than eight seconds. Such a requirement clearly impacts the design of the operational IT systems, but also the design of the analytical model.

The economical cost refers to the total cost of ownership and return on investment of the analytical fraud model. Although the former can be approximated reasonably well, the latter is more difficult to determine. Fraud analytical models should also be in line and comply with all applicable regulation and legislation with respect to, for example, privacy or the use of cookies in a web browser.

5. Experimental assessment

In this Section 5.1 we first describe the observed data set for the experiments. In Section 5.2 we present the experimental design and in Section 5.3 we show the results of the experiments.

5.1. Information about the real data set

We illustrate the proposed techniques on a data set that has been provided to our research group by a large European bank. The data set consists of fraudulent and legitimate transactions made with debit cards between September 2018 and July 2019. Note that the magnitude of the data set illustrated here is much smaller than data sets typically used in fraud prediction and its incidence of fraudulent transactions is also much higher. This is because a kind of white-listing (based on experience-driven business rules) was first applied to the data by the bank to filter out definitely safe transactions. The total data set contains 31,763 individual transactions, each with 14 attributes and a fraud label that indicates when a transaction is confirmed as fraudulent. This label was created internally in the bank by fraud investigators, and can be considered as highly accurate. Only 506 transactions in the data set were labeled as fraud, resulting in a fraud ratio of 1.6%.

The initial set of features include information regarding individual transactions, such as amount, timestamp, payment channel and beneficiary country. Table 5 contains examples of such typical attributes that are available for transactions.

Feature name	Description
Transaction ID	Transaction identification number
Timestamp	Date and time of the transaction
Originator's account number	Identification number of the originator's bank account
Beneficiary's account number	Identification number of the beneficiary's bank account
Beneficiary's name	Name of the beneficiary
Card number	Identification of the debit card
Payment channel	Electronic channel (e.g. online banking, mobile app, ...)
Authentication method	e.g. pin code, fingerprint, itsme, ...
Currency	Original currency (e.g. Euros, USD, ...)
Amount	Amount of the transaction in Euros
Originator country	Country from which the money is send
Beneficiary country	Country to which the money is send
Communication	Message provided with the transfer
Gender	Gender of the customer
Age	Age of the customer
Country	Customer's country of residence
Language	Customer's preferred language

Table 5: Examples of typical features of transactions.

5.2. Experimental design

In order to test the performance of machine learning models that only use these 14 initial features, we split the data into a training and testing set. Each one contains 70% and 30% of the transactions, respectively, stratified according to the fraud label to obtain similar fraud distributions as observed in the original data set. Table 6 summarizes the different data sets.

For the experiments we use the following popular classification methods: logistic regression (LR),

Set	Transactions	Frauds
Total	31,763	506
Training	22,234	354
Testing	9,529	153

Table 6: Summary of the data sets.

decision tree (DT), using the CART algorithm (Breiman et al., 1984), and gradient boosted trees (GBT), using the XGBoost algorithm (Chen and Guestrin, 2016). Logistic regression is often used in the industry because it is fast to compute, easy to understand and interpret. Moreover, logistic regression is often used as a benchmark model to which other classification algorithms are compared. Commonly used decision tree algorithms include CART (Breiman et al., 1984) and C4.5 (Quilan, 1993). The tree-like structure of a decision tree makes it particularly easy to gain insight in its decision process. This is especially useful in a fraud detection setting to understand how fraud is committed and work out corresponding fraud prevention strategies. XGBoost is short for eXtreme Gradient Boosting (Chen and Guestrin, 2016). It is an efficient and scalable implementation of the gradient boosting framework by Friedman et al. (2000) and Friedman (2001), but it uses a more regularized model formalization to control over-fitting, which gives it better performance. The name XGBoost refers to the engineering goal to push the limit of computational resources for boosted tree algorithms. The XGBoost algorithm is widely used by data scientists to achieve state-of-the-art results on many machine learning challenges and has been used by a series of competition winning solutions (Chen and Guestrin, 2016). Note that recent model explaining techniques, such as SHapley Additive exPlanation (SHAP,Lundberg and Lee (2017)) and Local Interpretable Model-agnostic Explanations (LIME, Ribeiro et al. (2016)) make it possible to provide model interpretability for such black box methods. These perturbation-based methods estimate the contribution of individual features towards a specific prediction. The purpose of this paper is to illustrate the benefit of the proposed data engineering techniques to the performance of fraud detection models regardless of the chosen model structure. Therefore, all three classifiers (LR, DT and GBT) are trained on the training set using their default parameters as suggested by their respective authors. The performance of the three classifiers is evaluated on the testing set using Precision, Recall (i.e. hit rate), F_1 measure, false positive rate (FPR, i.e. false alarm rate), Area Under Precision Recall Curve (AUPRC), Savings, and the fraction of fraudulent amounts that are detected. Hereby a decision threshold of $t = 50\%$ is used. For the calculation of the Savings measure, we choose a fixed cost of $c_f = 5$ Euros.

5.3. Results

Table 7 contains the performance of logistic regression (LR), decision tree (DT) and gradient boosted trees (GBT) on the testing set using the 14 original features (top). When we include RFM features and time features using the von Mises distribution, the performance of all three models improves significantly (middle of Table 7). In particular the Savings, F_1 and $AUPRC$ values of the three models have clearly increased. Their overall performance is further enhanced when we add

the features that are based on the anomaly detection techniques (bottom of Table 7). Using the original features, the three models are only able to detect around 50% of the fraudulent amounts. By including the features that are created by the various feature engineering methods, the improved models can block more than 70% of the stolen money and thus saving more than 67% of the costs compared to not using any fraud detection system.

Original features

	Precision	Recall	F_1	FPR	AUPRC	Savings	% of fraud amount detected
LR	0.6154	0.3810	0.4706	0.0025	0.4417	0.5117	0.5340
DT	1.0000	0.1905	0.3200	0.0000	0.3050	0.3191	0.3260
GBT	0.7778	0.3333	0.4667	0.0010	0.4632	0.5068	0.5223

Including RFM and other time-related features

	Precision	Recall	F_1	FPR	AUPRC	Savings	% of fraud amount detected
LR	0.5625	0.4286	0.4865	0.0035	0.4680	0.5483	0.5757
DT	0.8000	0.3810	0.5161	0.0010	0.4836	0.6635	0.6807
GBT	0.6923	0.4286	0.5294	0.0020	0.6333	0.5979	0.6202

Including features based on anomaly detection techniques

	Precision	Recall	F_1	FPR	AUPRC	Savings	% of fraud amount detected
LR	0.7647	0.6190	0.6842	0.0020	0.6975	0.6751	0.7042
DT	0.8125	0.6190	0.7027	0.0015	0.6370	0.6883	0.7158
GBT	0.8750	0.6667	0.7568	0.0010	0.7669	0.7908	0.8183

Table 7: Performance of logistic regression (LR), decision tree (DT) and gradient boosted trees (GBT) on the testing set using (top) the 14 original features, (middle) the RFM and other time-related features, (bottom) and the features based on anomaly detection techniques.

While the data set is now extended with new features, the imbalance between the fraudulent and legitimate transactions remains. To address this issue we apply the following over-sampling methods on the extended training set: SMOTE, ADASYN, MWMOTE and ROSE, each with their default parameters as suggested by their respective authors. We use these over-sampling techniques such that the new, re-balanced training set contains a ratio of 90% legitimate cases versus 10% fraud cases. In Table 8 we present the results for all three classifiers with each of the over-sampling methods. Notice how the performance varies depending on the chosen over-sampling method. The Savings value of the logistic regression model is mostly improved with MWMOTE as well as SMOTE and ROSE. The Savings value of the decision tree, however, only increases with ADASYN and SMOTE. While logistic regression and decision tree may benefit from over-sampling methods, the overall performance of the

gradient boosted trees is decreasing. This may be due to the boosting algorithm which could be over-fitting the classifier on the over-sampled training set resulting in a lesser performance on the testing set. Depending on the chosen classification method, there is definitely potential in over-sampling the training set with synthetic fraud cases, although there is not one over-sampling technique that will always yield the best result.

Logistic regression (LR)

	Precision	Recall	F_1	FPR	AUPRC	Savings	% of fraud amount detected
Original	0.7647	0.6190	0.6842	0.0020	0.6975	0.6751	0.7042
SMOTE	0.4103	0.7619	0.5333	0.0116	0.6408	0.7647	0.8316
ADASYN	0.4167	0.7143	0.5263	0.0106	0.6924	0.6674	0.7291
MWMOTE	0.4706	0.7619	0.5818	0.0091	0.6388	0.7733	0.8316
ROSE	0.4324	0.7619	0.5517	0.0106	0.6692	0.7681	0.8316

Decision tree (DT)

	Precision	Recall	F_1	FPR	AUPRC	Savings	% of fraud amount detected
Original	0.8125	0.6190	0.7027	0.0015	0.6370	0.6883	0.7158
SMOTE	0.5000	0.7619	0.6038	0.0081	0.5118	0.7712	0.8261
ADASYN	0.5667	0.8095	0.6667	0.0066	0.3716	0.7987	0.8501
MWMOTE	0.4545	0.7143	0.5556	0.0091	0.4001	0.6739	0.7305
ROSE	0.6190	0.6190	0.6190	0.0040	0.6565	0.6866	0.7226

Gradient boosted trees (GBT)

	Precision	Recall	F_1	FPR	AUPRC	Savings	% of fraud amount detected
Original	0.8750	0.6667	0.7568	0.0010	0.7669	0.7908	0.8183
SMOTE	0.6842	0.6190	0.6500	0.0030	0.7146	0.5941	0.6266
ADASYN	0.8462	0.5238	0.6471	0.0010	0.7763	0.5962	0.6184
MWMOTE	0.7500	0.5714	0.6486	0.0020	0.6931	0.5975	0.6249
ROSE	0.6667	0.0952	0.1667	0.0005	0.4341	0.0430	0.0482

Table 8: Performance of logistic regression (top), decision tree (middle) and gradient boosted trees (bottom) on the testing set using different over-sampling methods: SMOTE, ADASYN, MWMOTE and ROSE.

6. Conclusions and future research

In this paper, we extensively researched data engineering in a fraud detection setting. More specifically, we decomposed data engineering into feature engineering and instance engineering. Our motivation for doing so is that, based upon past extensive research, it it is our firm belief that the best

way to boost the performance of any analytical technique is to smartly engineer the data instead of overly focusing on the development of new, often times highly complex, analytical techniques giving us analytical models which are often only poorly benchmarked and give us no interpretability at all. We used a payment transactions data set from a large European Bank to illustrate the substantial impact of data engineering on the performance of a fraud detection mode. We empirically showed that both the feature engineering and instance engineering steps significantly improved the performance of popular analytical models. Moreover, we have illustrated that by clever engineering of the data simple analytical techniques as logistic regression and classification trees yield very good results.Although the focus in this paper is on payment transactions fraud, the discussed techniques are also useful or could be extended to other types of fraud, e.g. in healthcare, insurance or e-commerce.

References

Amin, A., Anwar, S., Adnan, A., Nawaz, M., Howard, N., Qadir, J., Hawalah, A., Hussain, A., 2016. Comparing oversampling techniques to handle the class imbalance problem: A customer churn prediction case study. IEEE Access 4, 7940–7957.

Angiulli, F., Pizzuti, C., 2002. Fast outlier detection in high dimensional spaces, in: European conference on principles of data mining and knowledge discovery, Springer. pp. 15–27.

Atkinson, A., Riani, M., 2000. Robust diagnostic regression analysis. Springer Science & Business Media.

Baesens, B., Höppner, S., Verbeke, W., Verdonck, T., 2020. Instance-dependent cost-sensitive learning for detecting transfer fraud. arXiv preprint arXiv:2005.02488 .

Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., Vanthienen, J., 2003. Benchmarking state-of-the-art classification algorithms for credit scoring. Journal of the operational research society 54, 627–635.

Bahnsen, A.C., Aouada, D., Stojanovic, A., Ottersten, B., 2016. Feature engineering strategies for credit card fraud detection. Expert Systems with Applications 51, 134–142.

Barabesi, L., Cerasa, A., Cerioli, A., Perrotta, D., 2018. Goodness-of-fit testing for the newcomb-benford law with application to the detection of customs fraud. Journal of Business & Economic Statistics 36, 346–358.

Barkan, O., Koenigstein, N., 2016. Item2vec: Neural item embedding for collaborative filtering. arXiv:1603.04259.

Barua, S., Islam, M.M., Yao, X., Murase, K., 2012. Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. IEEE Transactions on knowledge and data engineering 26, 405–425.

Bhattacharyya, S., Jha, S., Tharakunnel, K., Westland, J.C., 2011. Data mining for credit card fraud: A comparative study. Decision Support Systems 50, 602–613.

Boudt, K., Rousseeuw, P.J., Vanduffel, S., Verdonck, T., 2020. The minimum regularized covariance determinant estimator. Statistics and Computing 30, 113–128.

Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and regression trees. Wadsworth Int. Group 37, 237–251.

Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J., 2000. Lof: identifying density-based local outliers, in: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pp. 93–104.

Brito, M.R., Chávez, E.L., Quiroz, A.J., Yukich, J.E., 1997. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. Statistics & Probability Letters 35, 33–42.

Carroll, R.J., Ruppert, D., 1985. Transformations in regression: A robust analysis. Technometrics 27, 1–12.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. Smote: synthetic minority oversampling technique. Journal of artificial intelligence research 16, 321–357.

Chen, T., Guestrin, C., 2016. Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, ACM. pp. 785–794.

Dal Pozzolo, A., Caelen, O., Le Borgne, Y.A., Waterschoot, S., Bontempi, G., 2014. Learned lessons in credit card fraud detection from a practitioner perspective. Expert systems with applications 41, 4915–4928.

Davies, L., Gather, U., 1993. The identification of multiple outliers. Journal of the American Statistical Association 88, 782–792.

Davis, J., Goadrich, M., 2006. The relationship between Precision-Recall and ROC curves, in: Proceedings of the 23rd international conference on Machine learning, ACM. pp. 233–240.

European Central Bank, E., September 2018. Fifth report on card fraud. URL www.ecb.europa.eu/pub/cardfraud/html/ecb.cardfraudreport201809.en.html .

Fawcett, T., 2004. ROC graphs: Notes and practical considerations for researchers. Machine learning 31, 1–38.

Fawcett, T., 2006. An introduction to ROC analysis. Pattern recognition letters 27, 861–874.

Fernández, A., García, S., Galar, M., Prati, R.C., Krawczyk, B., Herrera, F., 2018. Learning from imbalanced data sets. Springer.

Fisher, N.I., 1995. Statistical analysis of circular data. cambridge university press.

Friedman, J., Hastie, T., Tibshirani, R., et al., 2000. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The annals of statistics 28, 337–407.

Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Annals of statistics , 1189–1232.

Goldstein, M., Uchida, S., 2016. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PloS one 11, e0152173.

Grover, A., Leskovec, J., 2016. node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 855–864.

Hamilton, W.L., Ying, R., Leskovec, J., 2017. Inductive representation learning on large graphs. `arXiv:1706.02216`.

Hand, D.J., Whitrow, C., Adams, N.M., Juszczak, P., Weston, D., 2008. Performance criteria for plastic card fraud detection tools. Journal of the Operational Research Society 59, 956–962.

He, H., Bai, Y., Garcia, E.A., Li, S., 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning, in: 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), IEEE. pp. 1322–1328.

Heritier, S., Cantoni, E., Copt, S., Victoria-Feser, M.P., 2009. Robust methods in biostatistics. volume 825. John Wiley & Sons.

Hubert, M., Vandervieren, E., 2008. An adjusted boxplot for skewed distributions. Computational statistics & data analysis 52, 5186–5201.

Jha, S., Guillen, M., Westland, J.C., 2012. Employing transaction aggregation strategy to detect credit card fraud. Expert systems with applications 39, 12650–12657.

Kovács, G., 2019. Smote-variants: A python implementation of 85 minority oversampling techniques. Neurocomputing 366, 352–354.

Krzanowski, W.J., Hand, D.J., 2009. ROC curves for continuous data. Chapman and Hall/CRC.

Lessmann, S., Baesens, B., Seow, H.V., Thomas, L.C., 2015. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. European Journal of Operational Research 247, 124–136.

Ling, C.X., Huang, J., Zhang, H., et al., 2003. AUC: a statistically consistent and more discriminating measure than accuracy, in: Ijcai, pp. 519–524.

Liu, F.T., Ting, K.M., Zhou, Z.H., 2008. Isolation forest, in: 2008 Eighth IEEE International Conference on Data Mining, IEEE. pp. 413–422.

Lunardon, N., Menardi, G., Torelli, N., 2014. Rose: A package for binary imbalanced learning. R journal 6.

Lundberg, S.M., Lee, S.I., 2017. A unified approach to interpreting model predictions, in: Advances in neural information processing systems, pp. 4765–4774.

Marazzi, A., Villar, A.J., Yohai, V.J., 2009. Robust response transformations based on optimal prediction. Journal of the American Statistical Association 104, 360–370.

Maronna, R.A., Martin, R.D., Yohai, V.J., Salibián-Barrera, M., 2019. Robust statistics: theory and methods (with R). John Wiley & Sons.

Ngai, E.W., Hu, Y., Wong, Y.H., Chen, Y., Sun, X., 2011. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. Decision support systems 50, 559–569.

Nigrini, M.J., 2012. Benford's Law: Applications for forensic accounting, auditing, and fraud detection. volume 586. John Wiley & Sons.

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

Provost, F., Fawcett, T., Kohavi, R., 1998. The case against accuracy estimation for comparing classifiers. 5th int, in: Conference on Machine Learning, San Francisco, Kaufman Morgan, pp. 445–453.

Quilan, J.R., 1993. C4.5: Programs for machine learning. Morgan Kaufmann Publichers, San Mateo .

Raymaekers, J., Rousseeuw, P.J., 2020. Transforming variables to central normality. arXiv preprint arXiv:2005.07946 .

Riani, M., 2008. Robust transformations in univariate and multivariate time series. Econometric Reviews 28, 262–278.

Ribeiro, M.T., Singh, S., Guestrin, C., 2016. " why should i trust you?" explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 1135–1144.

Rousseeuw, P., Perrotta, D., Riani, M., Hubert, M., 2019. Robust monitoring of time series with application to fraud detection. Econometrics and statistics 9, 108–121.

Rousseeuw, P.J., Driessen, K.V., 1999. A fast algorithm for the minimum covariance determinant estimator. Technometrics 41, 212–223.

Rousseeuw, P.J., Hubert, M., 2018. Anomaly detection by robust statistics. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8, e1236.

Rousseeuw, P.J., Leroy, A.M., 2005. Robust regression and outlier detection. volume 589. John wiley & sons.

Saerens, M., Latinne, P., Decaestecker, C., 2002. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. Neural computation 14, 21–41.

Saito, T., Rehmsmeier, M., 2015. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. PloS one 10.

Swets, J.A., 2014. Signal detection theory and ROC analysis in psychology and diagnostics: Collected papers. Psychology Press.

Tukey, J.W., 1977. Exploratory data analysis. volume 2. Reading, MA.

Van Vlasselaer, V., Eliassi-Rad, T., Akoglu, L., Snoeck, M., Baesens, B., 2017. Gotcha! network-based fraud detection for social security fraud. Management Science 63, 3090–3110.

Whitrow, C., Hand, D.J., Juszczak, P., Weston, D., Adams, N.M., 2009. Transaction aggregation as a strategy for credit card fraud detection. Data mining and knowledge discovery 18, 30–55.

Zhu, B., Gao, Z., Zhao, J., vanden Broucke, S.K., 2019. Iric: An r library for binary imbalanced classification. SoftwareX 10, 100341.