# Identifying fiscal fraud with anomaly detection techniques

Klara M. Gutekunst

University of Kassel, 34125 Kassel, Germany
klara.gutekunst@student.uni-kassel.com

**Abstract.** This survey addresses the task of identifying fiscal fraud using anomaly detection techniques with regard to the problem governments' tax offices are facing in terms of tax evasion. As it is difficult to find profound studies of this specific problem, papers about a transferable problem are used to identify different approaches to face the original problem. Multiple supervised methods to tackle credit card fraud are listed. The main focus of this survey is on unsupervised techniques. The four methods **ARIMA!**, **IF!**, **AE!**, and **SOM!** are described in more detail. This paper provides a discussion on how applicable each of these approaches is to the original problem, as well as a short outline of the approaches' shortcomings.

**Keywords:** unsupervised learning · anomaly detection · fiscal fraud · fraud detection

## 1 Introduction

Due to the fact, that a lot of monetary transactions happen via credit cards, the potential for fraudulent actions in this domain is increasing. Since these fraudulent actions result in great loss of money for credit card owners or companies, this topic has become a priority and many data scientists have started focussing their research on fiscal fraud detection. While detection of fraudulent credit card actions may happen in real-time, fiscal fraud detection in tax offices is offline[1]. Although tax evasion does not directly affect individuals, but the government, techniques of credit card fraud detection may be adapted to identify tax evasion. The goal of this survey is to outline four of the ways researchers have already found to identify financial fraud with unsupervised techniques.

The remainder of this article is structured as follows. First, the problem at hand is discussed in **??**. After that in **??**, the existing techniques for a transferable anomaly detection problem are briefly listed. In **??**, four unsupervised techniques are described in more detail. Afterwards, these techniques are compared in a discussion about their potential with regard to the problem at hand. Finally, there is a conclusion and outlook.

[1] According to [**gruhl2022**], an algorithm works in an offline fashion, if it requires access to the complete data set at once to perform a successful detection.

## 2 Problem

A government's tax office has various ways to detect fraudulent activities of companies that deliberately take actions to pay fewer taxes than they are supposed to pay. One way of detecting fraudulent activities is using the cash register data of companies to identify anomalies, which is currently performed by humans. Data mining techniques can support this process.

The cash register of a business is supposed to log every transaction and thus, is a valuable resource for analysing the business' finances. The data's structure depends on the cash register system. The documentation of an exemplary cash register's data structure is given in [**Vectron**]. Every transaction is logged with attributes such as `date`, `time`, `clerk`, `identification`, and `item`. When exporting the data it is partitioned and stored in different files which are linked via keys.

Some systems have data subsets that are redundant, due to the fact that there are data files considered fiscally ir-/relevant. In this context, it is important to investigate both data subsets considered fiscally ir- and relevant, since it is possible that some anomalies are only evident in files consisting of non-aggregated entries, which may be considered fiscally irrelevant according to the documentation of the cash register system.

The structures of data sets from different cash register systems usually have some similarities, since there are standards for cash register data set by the ***IFRS!*** (**IFRS!**) according to [**ff_review_techniques**]. One similarity in any cash register data set is periodic behaviour, for instance in features such as `time of transaction`. A hypothesis regarding the data is that there are similarities between weekday data over months.

## 3 State of the art

In this section, technical terms specific to the domain are defined and briefly discussed. Moreover, the structure of feasible credit card data is outlined. In the end, a brief overview of some of the financial fraud detection techniques is provided.

**Financial fraud:** Financial fraud may be defined by "any intentional act to deprive another of property or money by guile, deception, or other unfair means" (from [**ff_review_techniques**], based on [**acfe_ff**]). There are several different types of financial fraud, for instance, money laundering, credit card or insurance fraud as listed in [**ff_review_techniques**]. This survey's focus is on credit card fraud, since its detection techniques can be transferred to the problem presented in **??**. According to [**ff_review_techniques**] there are multiple types of credit card fraud including application fraud[2] and behavioural fraud (unauthorised usage of credit cards by third parties, bankruptcy fraud[3]).

[2] identity fraud to issue new cards from credit companies
[3] filing for personal bankruptcy instead of paying back the balance

**Anomalies:** There is no general definition of an *anomaly*. However, the authors of [**ff_review_techniques**] use the definition of Hawkins, which identifies outliers or anomalies as observations which deviate so much from the other observations as to arouse suspicions that they were generated by a different mechanism. There are three different types of anomalies, which are examined in [**gruhl2022**, **ff_review_techniques**].

*Point anomalies* are single instances of the data, which deviate from the rest of the data set. They can occur in any data set.

There are *contextual* and *behavioural* attributes. For instance, `time` in time-series data describes the context of that data and is therefore a *contextual* attribute. Regarding this example from [**ff_review_techniques**], the attribute `temperature` is a non-contextual attribute and thus, is a *behavioural* attribute. *Contextual anomalies* are individual instances which deviate in terms of the *behavioural* attribute in a specific context. For instance, the temperature of a certain month is exorbitant greater than the ones documented in the respective months in other years. These anomalies can only occur if there are contextual attributes in the data.

Moreover, Gruhl discusses *collective* anomalies or so-called *novelties*, which are multiple data instances, whose occurrence as a group is anomalous with respect to the whole data set, in [**gruhl2022**]. They can only occur if there is a relationship between the data instances.

**Challenges of Anomaly Detection:** Anomaly detection is challenging due to a variety of reasons pointed out in [**gruhl2022**, **ff_review_techniques**]. Firstly, there are numerous types of normal behaviourism and therefore, it is difficult to consider all of them when classifying an individual instance. Respectively, anomalous individuals may differ greatly in their characteristics as they are irregular. Moreover, the boundary between normal and anomalous individuals often lacks precision. It is difficult to determine, since the underlying model of normal data may change over time. Another issue when dealing with anomaly detection tasks is the lack of labelled data for training as a consequence of the costly and time-consuming nature of this task. Given that there is usually some degree of natural noise in the data, it is difficult to distinguish between noise and actual anomalies. One also has to consider changing the data set's dimension. However, certain relations or anomalous behaviours may be evident in a certain dimension but become hidden after a dimension reduction.

**Data labels:** The goal of anomaly detection is to assign a label to every instance of a data set indicating whether it is an anomaly or not. In some cases, there are already labels present for a portion of the data set. As stated in [**ff_review_techniques**], there are different types of anomaly detection techniques which either require labelled data (semi-/supervised) or not (unsupervised).

**Credit card data:** Credit card data may be stored as a file containing $n$ features, such as `time`, `amount`, `id`, for $m$ transactions. Each transaction $t_i$ has a unique id $i$. The data sets are highly imbalanced. There is data available online, for instance on kaggle, however, due to confidentiality issues, the data from [**kaggle_ex**] is not provided in its original form. In this case, ***PCA!*** (**PCA!**) transformation was used on all features except `time` and `amount`. Originally, credit card data may contain non-numeric features. Depending on the context, credit card data contains transactions of one or multiple individuals. Hence, in some cases, the data is considered a time series.

**Models and techniques:** Data preprocessing is important, since, for instance (supervised) machine learning techniques ignore minority classes in imbalanced data sets and thus, perform poorly on them.

According to [**ff_profiles**], cardholder profiles are created based on the cardholders' spending behaviours and patterns in order to detect credit card fraud. Abnormal transactions, such as drawing out unusually large amounts of money from inconvenient locations, may indicate fraudulent actions.

As stated in [**ff_review_techniques**], models used for identifying credit card fraud include ***DT!s*** (**DT!**), ***SVM!*** (**SVM!**), ***LR!*** (**LR!**) and ***kNN!*** (**kNN!**), which are outlined in [**sahin2010detecting**]. It is also possible to combine those models in an ensemble along the lines of the ***RF!*** (**RF!**) algorithm. Frequently used deep learning anomaly detection techniques and architectures are ***NN!*** (**NN!**), ***CNN!*** (**CNN!**), ***LSTM!*** (**LSTM!**), ***AE!s*** (**AE!**) and ***GAN!*** (**GAN!**). The models and techniques stated above are outlined in [**ff_review_techniques**].

The unsupervised financial fraud detection techniques ***IF!*** (**IF!**), ***AE!*** (**AE!**), ***ARIMA!*** (**ARIMA!**), ***SOM!*** (**SOM!**), which form the basis of this survey and are respectively described in [**liu2008isolation**, **cf_AE**, **fd_ARIMA**, **fd_SOM**, **credit_f_SOM**].

## 4 Main part

Since it is most applicable to the problem presented in **??**, this section only focuses on unsupervised anomaly detection techniques. The remainder of this paper concentrates on transferable problems, such as credit card fraud, because there are more scientific papers covering these problems.

### 4.1 Isolation Forest

The goal of this approach is to isolate anomalies from the normal data. The paper [**liu2008isolation**] describes and evaluates this approach using multiple different data sets. Based on the hypothesis that anomalies stand out with regard to the values of their attributes, they are identified by the small number of separations necessary to isolate them in the process of recurrent sub-sampling of the data set.

The concept of binary trees is essential for the technique ***IF!*** (**IF!**). Every node is either an external or internal node with no or respectively two children. Children of a node correspond to a partition of the parent node's data set.

A tree is constructed as follows:

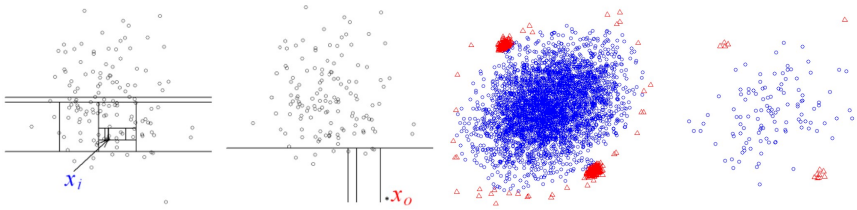
1. root of tree $\coloneqq$ data set $X$; $\psi \coloneqq$ sub-sampling size $\widehat{=} |X|$
2. A parent node $T$, has two children nodes $T_l, T_r$, a test attribute $q$ and a split value $p$
   (a) $q$ is randomly selected from existing attributes
   (b) $p$ is randomly selected within minimum and maximum value of $q$
   (c) $T_l \coloneqq \{x \mid x \in X \wedge x[q] < p\}$
   (d) $T_r \coloneqq \{x \mid x \in X \wedge x[q] \geq p\}$
3. Recursively divide $X$ into sub-samples $X_i$, until either
   (a) The tree reaches the height limit $l \coloneqq \lceil \log_2 \psi \rceil$
   (b) $|X_i| = 1$
   (c) All data in $X_i$ have the same values

According to [**gruhl2022**, **liu2008isolation**] the characteristics of the constructed trees are:

1. $n \coloneqq$ number of external nodes of an ***iTree!*** (**iTree!**)
2. $n - 1$ internal nodes in an **iTree!**
3. $h_i(x) \coloneqq$ number of edges from root node to external node $x \widehat{=}$ Path length in **iTree!** $i$
4. $t \coloneqq$ total number of **iTree!**s
5. $\bar{h(x)} \coloneqq \frac{1}{t} \sum_{i=1}^{t} h_i(x) \widehat{=}$ average path length of $x$
6. $c(\psi) \coloneqq$ average path length of a binary tree for a given sample set of $\psi$ observations
7. $s(x, \psi) \coloneqq 2^{-\frac{\bar{h(x)}}{c(\psi)}} \widehat{=}$ anomaly score
   (a) $\bar{h(x)} \to c(\psi) \Rightarrow s \to 0.5$
   (b) $\bar{h(x)} \to 0 \Rightarrow s(x, \psi) \to 1 \widehat{=} x$ is an anomaly
   (c) $\bar{h(x)} \to n - 1 \Rightarrow s(x, \psi) \to 0 \widehat{=} x$ is a normal instance

The approach is based on the hypothesis, that anomalies are less frequent (a minority in the data set) and have attribute values that extremely deviate from those of normal instances. Based on this hypothesis, the authors of [**liu2008isolation**] conclude, that anomalies, as depicted in **??**, require fewer partitions to be isolated than normal instances of a data set, which is visualised in **??**. The number of partitions required to isolate a data point is equal to path length $h_i(x)$ in an **iTree!** $i$. Hence, an average path length of $\bar{h(x)} \to 0$ suggests that the grand majority of the **iTree!**s has short paths from the root to the instance $x$ and thus, consider $x$ an anomaly.

Anomalies are identified as nodes having shorter path lengths, which means they are isolated earlier than normal instances. Hence, the process of dividing the data set into sub-samples can be terminated at the tree height limit $l$, since the anomalies should be isolated by then. However, not all normal instances are isolated at this point and thus, a partial model is created. If a data point $x$




(a) Normal instance (b) Anomaly $x_0$ (c) Original data (d) Sub-sample

Fig. 1: Visualisation of instance isolation and sub-sampling from [**liu2008isolation**]. The normal instance $x_i$ in **??** requires more partitions (visualised by horizontal and vertical lines) to be isolated than the anomaly $x_0$ in **??**, due to $x_0$'s solitary position. In **??** for the original data and **??** for a sub-sample of the original data, blue circles denote normal instances, whereas red triangles denote anomalies. Anomalies are isolated quicker in less dense data sets such as **??**.

belongs to the part of the tree, which was not built, because for instance, the tree height limit $l$ has been reached and thus, expansion was terminated, an adjustment account $c$(size of unbuilt subtree) is added when calculating its path length.

Since **iTree!**s work well if the sampling size is kept small, sub-sampling is conducted. Comparing **??** and **??**, it becomes clear that fewer data points are favourable to the isolation technique since they are less dense and thus, anomalies are more likely to be isolated early.

According to [**liu2008isolation**], swamping refers to wrongly identifying normal instances as anomalies. Masking is the existence of too many anomalies concealing their own presence. Both problems arise from too much data. **iTree!**s build a partial model by sub-sampling and thus, control the data size and specify on a sub-sample of anomalies or respectively a lack of them.

Anomaly detection using **IF!**s is a two-stage process. As stated in [**gruhl2022**, **liu2008isolation**], first, in the training stage, sub-samples of size $\psi$ from the training set are used to build **iTree!**s. Then, in the evaluation stage, test instances are passed through the **iTree!**s to obtain anomaly scores based on their path length. The highest $m$ anomaly scores correspond to the $m$ anomalies identified by the **IF!**.

The input parameter $\psi$ controls the input data size in the training stage whereas the number of trees $t$ controls the ensemble size. Empirical studies from [**liu2008isolation**] suggest that $\psi = 2^8 = 256$ and $t = 100$ are optimal to perform anomaly detection across a wide range of data. Owing to the fact that there are multiple **iTree!**s it is called a forest. Obviously, multiple **iTree!**s have different sets of partitions, due to the fact that they are randomly generated.

### 4.2 Autoencoder

The idea of the approach is to use a low-dimensional version of the data to identify anomalies either using a trained model or the reconstructing error when reconstructing the encoded data. The approach from [**cf_AE**] consists of two stages. Firstly, the dimension of the high-dimension credit card data is reduced using an undercomplete ***AE!*** (**AE!**), which is a feed-forward neural network depicted in **??** that learns efficient (non-correlated) encodings for the input data. It is called undercomplete because the dimension of its hidden layer, or so-called latent space, is smaller than the dimension of the input layer. Feed-forward means that after training the information moves in only one direction (forward) and thus, no loops or cycles exist. However, while training, the network employs backpropagation[4] to update the parameters.

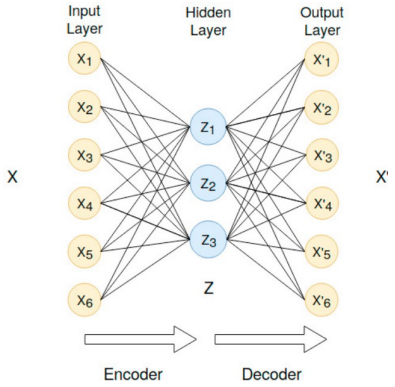



Fig. 2: Structure of **AE!** from [**cf_AE**]

The goal of an **AE!** is to approximate the identity function $f_\theta(X) = X$ (trivial solution eliminated) for input $X$ and function parameters to be learned $\theta$. The input and output layer have the same dimension since the encoded data is reconstructed in the output layer.

The mathematical background of the encoder is given in (**??**). The dimension reduction is computed using weight $W_\theta$, bias $B_\theta$ of the encoder layer, and the (non-)linear encoder activation function $f_E(.)$. Owing to the usage of a non-linear activation function, the neural network is capable of more than linear regression.

$$Z = f_E(W_\theta X + B_\theta) \tag{1}$$

The decomposition of the reduced dimensionality data to the output $X'$ is calculated analogously to (**??**). (**??**) is used for the decoder.

$$X' = f_D(W_\theta Z + B_\theta) \tag{2}$$

[4] updating weights and biases of the neurons based on forward propagation error

The Euclidean distance from (**??**) between the input $X$ and the output $X'$, which is reconstructed from $Z$, is considered the reconstruction error. According to [**AE_RF**], error backpropagation and gradient descent[5] are employed to train the **AE!** with the goal of minimising the reconstruction error. The reconstruction error can be used to identify anomalies: If the reconstruction error is greater than a certain threshold[6], $X$ is considered an anomaly based on the hypothesis that the **AE!** has approximated the identity function for normal instances.

$$\triangle(X, X') = \| X - X' \|_2 \tag{3}$$

To avoid overfitting, a regulariser tuning the objective function of the learning algorithm may be added.

The approach from [**cf_AE**] first uses the encoder, which transforms $X$ to $Z$. Since the authors assume that $X$ can be reconstructed using $Z$, it is a valid representation of the input transaction. Then, a classifier is trained on $Z$. In [**cf_AE**] labelled transactions of a training set are used to train a ***MLP!*** (**MLP!**), ***kNN!*** (**kNN!**), ***LR!*** (**LR!**), but generally, every model can be used to identify anomalies.

**Algorithm 1** ***AE-PRF!*** (**AE-PRF!**)

**Input:** training data $D_{train}$, validation data $D_{val}$, test data $D_{test}$, metric $M$
**Output:** classification of each instance (0 if normal, 1 else)

Train the **AE!** model $\textbf{AE!}_T$ with $D_{train}$ ▷ pseudocode based on [**AE_RF**]
$T \leftarrow \textbf{AE!}_T(D_{train})$ ▷ $T$ set of training data after dimension reduction
Train the **RF!** model $\textbf{RF!}_T$ with $T$
$V \leftarrow \textbf{AE!}_T(D_{val})$ ▷ $V$ set of validation data after dimension reduction
**for** $\theta \leftarrow 0$ `to` $1$ `step` $0.01$ **do** ▷ iterative testing of possible threshold values $\theta$
    **for** $v \in V$ **do**
        $p \leftarrow \textbf{RF!}_T(v)$ ▷ probability $p$ of fraud classification
        **if** $p > \theta$ **then**
            `result`$[\theta][v] \leftarrow 1$
        **else**
            `result`$[\theta][v] \leftarrow 0$
Find best $\theta*$ by comparing all *result* values in terms of metric $M$
$C \leftarrow \textbf{AE!}_T(D_{test})$ ▷ $C$ set of test data after dimension reduction
**for** $c \in C$ **do**
    $q \leftarrow \textbf{RF!}_T(c)$
    **if** $p > \theta*$ **then**
        `output`$[c] \leftarrow 1$
    **else**
        `output`$[c] \leftarrow 0$
**return** *output*

[5] iterative algorithm to find the function's coefficients that minimise the corresponding cost function using derivatives and the learning rate
[6] e.g. iterative computation of the threshold value analogously to **??**

[**AE_RF**] proposes an ***AE-PRF!*** (**AE-PRF!**), which is the combination of an **AE!** to extract transaction data features and a **RF!** with probabilistic classification to assign either the label fraudulent or normal to credit card transactions. The approach **AE-PRF!** is illustrated in pseudocode in **??**. The **RF!** in the paper is an ensemble learning model consisting of **DT!**s and employs bagging[7]. In this case the **DT!**s' internal nodes split the data set on the optimal (i.e. largest information gain) value of the corresponding split attribute. The splitting terminates if either all data of a node have the same value, the number of instances at a node reaches a predefined minimum limitation or the depth of the node reaches a predefined maximum limitation. An instance is then passed through the tree and classified by an individual tree based on the label of its corresponding leaf. The final **RF!** classification of that instance is the result of a majority voting or averaging of the individual **DT!**s, which have been constructed prior by using bagging. The probabilistic classification requires an additional parameter threshold $\theta \in [0, 1]$. The **RF!** model classifies an instance as fraudulent with probability $p$ and respective as normal with probability $1 - p$. The final **AE-PRF!** classification of that instance is fraudulent if $p > \theta$ is true and normal otherwise. The optimal threshold $\theta*$ can be determined by iterative testing of the classification performance in terms of a specific metric $M$ on different values for $\theta$.

### 4.3 Autoregressive integrated moving average

Based on the hypothesis that the `daily count of transactions` for a given customer for a certain point in time is correlated to some lagged values before that point in time, a model is trained using lagged values. Instances that deviate greatly from the predicted value of the trained model are considered fraudulent. The approach ***ARIMA!*** (**ARIMA!**) proposed in [**fd_ARIMA**] is the combination of an ***AR!*** (**AR!**) model and a ***MA!*** (**MA!**) model. It frames the problem of credit card fraud detection as an anomaly detection task in time series, where the variable presented as the time series is the `daily count of transactions` for a given customer.

An **AR!** model assumes the value $X_t$ for time unit $t$ is based on the correlated lagged (i.e. previous) observations $X_{t-i}, 0 < i \leq p$, a constant $c$ and an error term $\omega_t \sim N(0, \sigma^2)$ as stated in (**??**).

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \omega_t \tag{4}$$

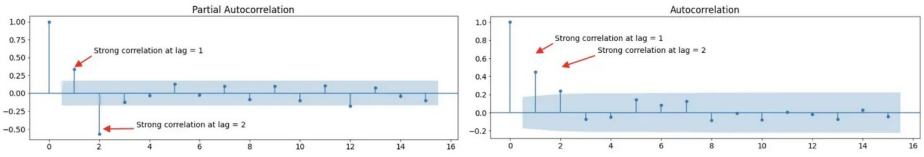
The coefficients $\phi = (\phi_1, ..., \phi_p)$ are estimated using the maximum likelihood estimation process. The number of lagged observations $X_{t-i}$ to consider (i.e. the ones statistically relevant) is indicated by $p \in \mathbb{N}$, which is calculated using

[7] independently training multiple individual **DT!**s based on random subsets of data and attributes

***PACF!*** (**PACF!**).

$$C(X,Y) = \sum_i \frac{(X_i - \mu_X)(Y_i - \mu_Y)}{\sigma_X \sigma_Y} \tag{5}$$

**PACF!** is graphically visualised in **??** by plotting the autocorrelation (Pearson correlation of $X_t$ with lagged observation of $X$ calculated in **??** from [**p_acf**]) value on the y-axis and the lags ($t-i, 0 \leq i$) on the x-axis. A data point displayed is the direct correlation of $X_t$ and $X_{t-i}$ not taking into account the lagged observations $X_{t-j}, 0 < j < i$: $C(X_{n+k}, X_n | X_{n+k-1}, ..., X_{n+1})$. The correlation values of lagged observations outside a certain threshold (illustrated by the blue confidence intervals) are considered statistically relevant.




(a) **PACF!** of **AR!**(2) (b) **ACF!** of **MA!**(2)

Fig. 3: Plots of two time series from [**P_ACF**]. Values outside of the blue confidence interval are considered non-zero and thus, statistically relevant. **??**, **??** indicate respectively $p = 2$, $q = 2$.

A **MA!** model assumes the value $X_t$ for time unit $t$ is based on the correlated lagged (previous) errors $\omega_{t-j}, 0 < j \leq q$, the mean of the series $\mu$ and an error term $\omega_t \sim N(0, \sigma^2)$ as stated in (**??**).

$$X_t = \mu + \sum_{j=1}^{q} \theta_j \omega_{t-j} + \omega_t \tag{6}$$

The calculation of the coefficients $\theta = (\theta_1, ..., \theta_q)$ is analogue to **AR!**. This model assumes that the observations are located around the mean $\mu$, hence that the time series is stationary ($\mu, \sigma$ constant over time). Thus, (**??**) without $\omega_t$ predicts $X_t$ based on the mean of the series and taking certain lagged errors $\omega_{t-j}$ into account. The error $\omega_t$ is added to calculate the real observation. The number of statistically relevant lagged errors $\omega_{t-j}$ is indicated by $q$, which is calculated using an ***ACF!*** (**ACF!**). **ACF!** is graphically visualised in **??** by plotting the autocorrelation value on the y-axis and the lags ($t - i, 0 \leq i$) on the x-axis. It is important to note that a data point displayed is the indirect and direct correlation of $X_t$ and $X_{t-i}$ calculated using **??**. Lagged observations considered statistically relevant are identified as stated above.

$$X_t = c + \omega_t + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j \omega_{t-j} \tag{7}$$

The ***ARMA!*** (**ARMA!**) model in (**??**) is a combination of the **AR!** (**??**) and the **MA!** (**??**) models. The components are computed as stated above. However, the assumption of the **MA!** model, a time series is stationary, is not always true. Therefore, an **ARIMA!**$(p, d, q)$ model is constructed using the **ARMA!** model and the possibility of applying differencing of degree $d$ to data points to make them stationary (if necessary).

The Box-Jenkins method is used to tune the **ARIMA!** model. Firstly, the ***ADF!*** (**ADF!**) test verifies if the time series is stationary. The **ADF!** test rejects either hypothesis $H_0 : \phi_i = 1 \Rightarrow$ not stationary or $H_1 : \phi_i < 1 \Rightarrow$ stationary. Secondly, $p, q$ are configured as stated above. Then, the training phase estimates coefficients $\phi, \theta$ as stated above.

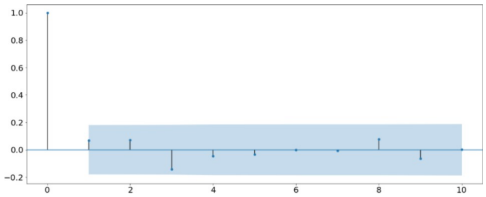



Fig. 4: Correlogram of residuals from [**fd_ARIMA**]. A correlogram is the visualisation of the correlation of $X_t$ with lagged observations of $X$. Since none of the lagged residuals is considered statistically relevant, the model selected is optimal.

Lastly, the model is evaluated. For an ideal model, the error is normally distributed with mean $\mu$ of 0, constant variance $\sigma$ and the residuals are stationary (no temporal structure) as illustrated in **??**.

In order to detect fraudulent credit card transactions, the time series is split into a training (preferably only containing legitimate transactions) and testing set. The **ARIMA!** model is trained applying the Box-Jenkins method to the training set.

$$z = \frac{x - \mu}{\sigma} \tag{8}$$

The fraudulent transactions of the testing set are identified by comparing the Z-score from (**??**), $x$ being the difference between predicted and actual `daily count of transactions`, $\mu, \sigma$ being the mean and variance based on the errors in predictions of the training set, with a certain threshold. A Z-score higher than the threshold corresponds to a fraudulent day.

### 4.4 Self-organising map

The goal of this approach is to identify fraudulent user accounts (i.e. bank accounts of individuals) using projection onto a two-dimensional feature space and a dissimilarity value.

The approach proposed in [**fd_SOM**] is ***SOM!*** (**SOM!**) as an example of artificial neural network architecture. It has a feed-forward topology and unsupervised self-organising training algorithm discussed in [**credit_f_SOM**].

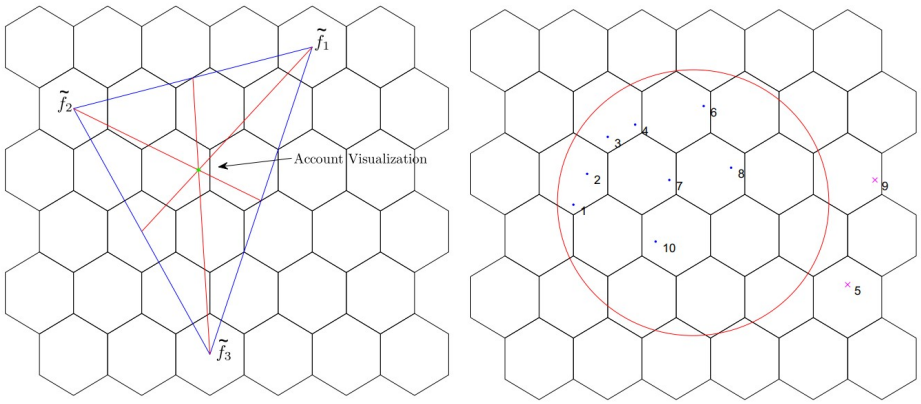
A record is a single transaction, which can be visualised as a vector (cf. single row in (**??**)). An entry of (**??**) corresponds to the value of the feature. A user account is a set of records from one specific user. It is represented by the data matrix from (**??**). $m$ is the number of features of a record and $n$ is the number of records, with $n \gg m$. Hence, $f_{i,j}$ is the $i$th value of the $j$th feature.

$$\begin{pmatrix} f_{1,1} & \dots & f_{1,m} \\ \dots & \dots & \dots \\ f_{n,1} & \dots & f_{n,m} \end{pmatrix} \tag{9}$$

Unlike other financial fraud detection techniques, this approach uses data visualisation to identify fraudulent user accounts rather than individual transactions within a user account.

Since **SOM!**s use vectors as input data, the matrices corresponding to a user account have to be transformed. This transformation or so-called projection of all features onto the **SOM!** is a dimension reduction $p$, where every original (high-dimensional) feature $f_j$ (with all inner-dimension entries $f_{i,j}, i \in [1, n]$, $n$ is number of records) is projected on a (2-dimensional) $\tilde{f}_j$ (**??**).

$$p : \mathbb{R}^n \to \mathbb{R}^2, f_j \mapsto \tilde{f}_j \tag{10}$$




(a) User account representation (b) Threshold-classification of instances

Fig. 5: **SOM!** visualisation from [**fd_SOM**]

The $i$th user account is represented by the centroid $c_i$ on the **SOM!** grid, which is calculated in (**??**) and visualised in **??**. There can be accounts that do

not correspond to a particular neuron (i.e. hexagon) in a **SOM!** grid. It is also possible that multiple similar accounts share a neuron.

$$c_i = \frac{1}{m} \sum_{j=1}^{m} \tilde{f}_{i,j} \tag{11}$$

According to [**fd_SOM**], the reduction of the inner-dimensionality of features from (**??**) does not reduce the amount of information, since the majority of records of a fraudulent user account are fraudulent themselves.

Each entry of the U-matrix of a **SOM!** corresponds to a neuron in the **SOM!** grid. Its value is the average dissimilarity between the neuron and its neighbours. Hence, a so-called ridge (sequence of high values) in the U-matrix represents a borderline separating the data clusters in the corresponding **SOM!** grid. **??** is a visualisation of an U-matrix. The colour of a hexagon corresponds to the value of the neuron in the U-matrix (cf. legend on the right side). Red coloured hexagons correspond to the highest values of average dissimilarity of neurons in the U-matrix. Hence, the group of red hexagons is a ridge. Although the closest user accounts below the ridge are considered non-fraudulent, the user accounts with more distance to the ridge (and below it) are marked as fraudulent. Thus, the ridge in fact separates the two clusters.

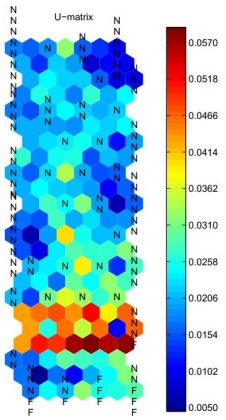



Fig. 6: Visualisation of U-matrix from [**fd_SOM**]. The letter $N$ marks non-fraudulent user accounts, whereas $F$ marks fraudulent user accounts. The coloured hexagons correspond to the values of the average dissimilarity of neurons and their neighbours in the U-matrix.

As stated in [**fd_SOM**], the classification of (non-)fraudulent instances is performed by a threshold-type binary classification technique. Visually, the threshold of the circle illustrated in **??** with centre $c$, which is the centroid of the entire

**SOM!** grid and radius $\tau$, which is calculated in (**??**). $v_{max}$ is the **SOM!** neuron corresponding to the maximal value in the U-matrix of the **SOM!** and $d(.,.)$ is a chosen dissimilarity measure. All points outside the circle are considered fraudulent user accounts and respectively all points within the threshold are considered normal user accounts.

$$\tau = d(c, v_{max}) \tag{12}$$

$$\varphi(a_i) = \begin{cases} \text{true} & \text{for } d(c_i, c) > \tau \\ \text{false} & \text{for } d(c_i, c) \leq \tau \end{cases} \tag{13}$$

(**??**) classifies a user account $a_i$ as either fraudulent (return value true) or non-fraudulent (return value false).

## 5 Conclusion and Outlook

The approach **IF!** described in [**liu2008isolation**] appears to be adaptable to the problem outlined in **??**. However, due to the fact that the data set has many (irrelevant) attributes, it is possible that the technique is time-consuming if the choice of split features is bad. To overcome this issue, a prior feature selection would be appropriate.

The issues of the approach from [**cf_AE**] using a two-stage technique including an **AE!** are that the second stage models proposed are using labelled data and that the model has to be retrained periodically because it does not adapt to changes in the pattern of fraud. The first problem can be resolved because the authors state that any classifier (i.e. an unsupervised one) or the reconstruction error can be used instead. The second issue is only a problem if the data from **??** contains evolving different types of fraud. Applying the method to multiple sub-samples of the data could be a solution.

The additional **RF!** after the **AE!** proposed by [**AE_RF**] to form the **AE-PRF!** is an example of the usage of an **AE!**. However, the quality of the threshold $\theta*$ strongly depends on the choice of metric. It is possible that human resources are required in order to provide a suitable metric, due to the fact that there is no labelled data available.

A problem of **ARIMA!** stated by the authors of [**fd_ARIMA**] is that **ARIMA!** assumes the data comes from observations equally spaced in time. However, transaction times of the problem outlined in **??** are unequally spaced. This issue can be solved using aggregated data, namely, aggregating data according to their dates creating equally spaced data. Moreover, the approach requires training on a subset of exclusively legitimate transactions which in this case cannot be provided. Furthermore, the specific approach identifies anomalies solely on the number of transactions conducted and thus, only uses a fraction of the features present in the data from **??**.

As stated in [**fd_SOM**], **SOM!** can be used as a clustering technique in order to detect fraudulent accounts. Using the representation of a reference user account (typical traits and behaviours) as the centre of the circle illustrating

the threshold could possibly further improve the approach. However, this approach may not be very applicable to the problem at hand, because it identifies whole user accounts as fraudulent based on the assumption that the majority of individual transactions classify likewise to the classification of the whole user account, which differs from the problem outlined in **??**.

Taking all the points stated above into account, the **IF!** appears to be the most applicable approach to face the problem at hand. However, with slight alterations, the other unsupervised learning methods seem to be valuable means to tackle the task of tax evasion detection. It may even be desirable to consider ensemble techniques using the methods discussed. For instance, first **AE!** may be used for a dimension reduction and then **IF!** can be applied to isolate the anomalies.

Advanced research with regard to these anomaly detection techniques may include comparing the approaches in terms of performance.