Generating process anomalies using a taxonomy of fraud characteristics and Markov models for accurate detection

Jochem Veldman
First supervisor: Dr. Ir. X. (Xixi) Lu
Second supervisor: Dr. G.C. (Inge) van de Weerd
Company supervisor: M.G.J. (Martijn) Heinen Msc.
Second company supervisor: J. (Joost) Ras Msc.

Business Informatics
Graduate School of Natural Science
Utrecht university
The Netherlands
August 10, 2022

Abstract

Supervised models perform much better than unsupervised models in detecting anomalies accurately. However, supervised models can often not be chosen since there is a lack of labelled data. Labelling data is a time-consuming and expensive task, so alternatives are wanted. Researchers have developed ways to work around the lack of labels, such as under- and oversampling. However, these methods are based on assumptions, the models trained with the generated data do not always show promising results because they are more likely to be over- or underfit, and the generation method leaves little room to tailor the generated anomalies to specific needs. In this research, we establish a taxonomy of anomaly characteristics from the domain of process mining and more specific characteristics from other domains. We have constructed three patterns from this taxonomy, namely, repetition, direct-follow and advanced repetition patterns, that can be used for generating anomalies. The data has been transformed into a Markov model, which allows easy incorporation of the designed patterns. The method is tested with a publicly available dataset from the UWV that contains click data from users on their website. For each pattern, we change the quantity of injected anomalous sessions and the deviation rate of the injected anomalies. The case study results show good performance for the models trained with all three patterns. Adding more anomalies to the training data increases AUC and recall but decreases precision. Changing the deviation rate of the anomalies has no significant impact on the repetition and direct-follow pattern but decreases the performance of the advanced repetition pattern. The taxonomy and the approached method can be generalized to other domains to counter the lack of labelled data.

Contents

List of Figures

List of Tables

1 Introduction

In 2018 the Dutch bank ING received a fine of 775 million euro because of their compliance issues with the Money Laundering and Terrorist Financing Prevention Act [48]. Three years later, the Dutch bank ABN AMRO received a fine of 480 million euros for the same reason [5]. Both banks neglected their role as gatekeepers of the financial system, and the information they stored about their clientele was not accurate enough. Because of neglecting their role as gatekeepers, they could not detect all anomalous transactions of their clients, and as a consequence of this, the banks hosted bank accounts that could freely be used for money laundering.

The story about the two banks illustrates the role that anomaly detection can play. With better performing anomaly detection, the banks could have done more against the money laundering. Anomaly detection is an essential task in data analytics used across a broad range of industries and suited for a wide variety of tasks [9, 37]. Besides the example of ING and ABN AMRO, anomaly detection is applied in intrusion detection, bot detection, fake review detection, terrorist activities and medical diagnosis [12].

An *anomaly* can be defined as a data point that does not match the expected behaviour or pattern [23]. It is a black swan between a group of white swans. An anomaly is of interest for the data analysis because it contains valuable and actionable information. For example, in the case of ING and ABN AMRO, detecting anomalies gives information about potential money laundering users. The bank can act on this information and, by doing so, prevent potential financial loss. In this research, an anomaly is seen as something different from noise. Noise is considered as hindrance and not valuable for the data analysis. E.g., a data instance with a temperature value of -10 Kelvin can be considered noise since the lowest possible temperature is 0 Kelvin.

Detection of anomalies is categorized into the following three approach: supervised, semi-supervised and unsupervised [37]. Supervised learning can be applied when there are labels for both normal and abnormal instances. Semi-supervised learning uses only labelled data of normal instances. As of last, unsupervised learning does not use any labels.

Supervised learning performs typically better than unsupervised learning, and if data is available, this should be the preferred method to use [39]. Labelled data, however, is limited available. The labels are not known in advance, and labelling the instances is very expensive and time-consuming [55, 65]. In this thesis, we are going to explore the possibilities to counter this slow and expensive process by proposing an alternative for obtaining labelled data.

In the rest of this chapter, we first discuss the problem statement. The problem statement provides an overview of what has been done in data generation as described in the scientific literature and what is still missing. The problem statement is followed by the relevance and contribution section, which explains why it is relevant to close the gap in the literature and how this research contribute to it. Next, we present the research questions used in this research. This is followed by the thesis outline that gives an overview of all the chapters in this research. As of last, a section is dedicated to the partner organization that supports conducting this research.

1.1 Problem statement

Labelled data is challenging to obtain, and labelling data instances is expensive and time-consuming. In domains such as healthcare, banking, or insurance, highly trained experts are needed to manually determine if the instance is an anomaly [88]. In addition, anomalies are infrequent, e.g. most people are real users and not fraudsters. The combination of these two factors means that, if labelled anomalies are available, they usually account for only 1% of the data [71, 8, 7]. Very few labelled anomalies result in imbalanced classes.

Imbalanced data is not very effective for training good supervised learning models [50]. An existing solution in the literature to counter this imbalance is to undersample the majority class or oversample the minority class [41, 43, 57]. However, this is not always an appropriate solution to the problem, as using these data generation techniques makes it more likely that the model trained on these data will under- or overfit [83]. Another problem is that if the data is available, companies are not always willing or legally able to give their data to researchers [75, 56, 99]. All in all, there is not always an easy solution to the missing labels, and the problem of lacking labelled data for training is widely faced in the scientific community.

Researchers each handled the lack of labelled data in their own way. In the remaining of the problem statement a number of methods to counter the lack of labelled data are discussed. In the literature section the methods and their limitations are discussed in more detail.

Wang et al. [93], and Gilani et al. [36] have manually labelled the data instances themselves. The research of Wang et al. and Gilani et al. shows good results, but a limitation of the method is the lack of scalability. Sun et al. [87, 86] has used unsupervised models, but unsupervised models tend to be less effective than the supervised methods. Other researchers [81, 44] have a small portion of labelled data available. They cluster all the data, and the data points in the clusters that include a labelled anomaly are all labelled as anomalies. Viswanath et al. [91] applies a similar method but uses a Principal Component Analysis as a clustering technique.

Other scientists have used data generation to counter the problem of missing labels. Alizadeh et al.[11] has generated a synthetic dataset by adding random noise to the data and using a real-life dataset that is on another topic than their research domain. Pegararo and van der Aalst [73] also adds random noise to an event log to create anomalies, whilst Cui et al.[27] uses a Monte-Carlo simulation to generate a dataset with anomalies. Van der Aalst [2] and Pourbafrani et al. [76] uses Discrete-Event Simulation (DES). For this method, a process model is made, and a program follows each step with the help of pre-defined rules. Alves de Medeiros and Günther[29] describes a widely used model in the process mining domain, namely Coloured Petri Nets (CPN). CPN can, just like DES, be considered as a rule-based generator. The CPN method is also used by Sahlabadi et al. [82] and Accorsi and Stocker [6]. These researchers are just a few examples, but the methods they use are quite comprehensive for all the research on the topic of data generation in general and in more depth for event log generation.

Different generation methods and labelling strategies of anomalies have been used in all this research. However, to the best of our knowledge, no research or public project uses domain knowledge and fraud characteristics to generate anomalies.

1.2 Relevance and contribution

The relevance and contribution of this research are high. The relevance of the research lies in the problem-solving capacity for problems occurring in the academic world and professional practice. The biggest challenge is that there is a lack of labelled data [15]. This lack of labelled data makes supervised models, which are the best method for anomaly detection, unable to be trained. Labelling data is time-consuming and expensive because highly trained, and well-paid domain experts mainly conduct it. In addition to solving the lack of labelled data, there are also positive side effects that make the generation of anomalies very relevant. With the help of generated anomalies, domain experts can generate anomalies that they anticipate to happen in the future. By generating the anomalies, the model is suited to detect anomalies that did not yet occur in the actual domain [99]. Another side effect is that process mining anomaly detection methods, in comparison with traditional anomaly detection methods, are more ethics friendly. Process mining anomaly detection only looks at the behaviour of instances and not at personal characteristics. In addition to the lack of labelled data, there is also a lack of an overview of anomaly characteristics in the scientific literature. This overview can give better insights into anomalies from different domains and can be used to adjust detection models.

The contribution of this research is threefold. First, the research investigates described anomaly characteristics in the scientific literature and combines these characteristics in a taxonomy. In addition to this taxonomy, the process mining characteristics are linked to the characteristics from the specific domains, which also adds to the contribution of this research. The second contribution is that this research explores how process mining anomalies can be generated and how best to do so. The last contribution of the research is testing the effect of the generated anomalies on the model performance. This last contribution follows the conclusion of whether generated anomalies can be helpful or not.

1.3 Challenges of anomaly generation

There are four major challenges when generating anomalies. These are: accurately mimicking realistic anomaly behaviour, generating diverse anomaly behaviour, the evolution of anomaly behaviour, and selection bias.

The first challenge for anomaly generation is that the generated anomalies mimic reality [2]. Only when realistic anomalies are generated it is helpful to train models on them. Otherwise, it is impossible to deploy the models, trained on the generated anomalies, in a real environment where they have to detect anomalies. In the machine learning domain this is also known also garbage in, garbage out. Next, it is a prerequisite that the generation process has to be performed smoothly and efficiently. It may be more economical to manually label data instances rather than generate them if it takes too much time.

The second challenge is generating diverse anomaly behaviour. The trained models on the generated anomalies should detect all anomalies in the domain in which they operate. Catching all anomalies has to be done to prevent any potential damage to the domain and at the same time avoid false positives, i.e., flagging genuine cases as anomalies. After a model detects an anomaly, a human has to investigate the anomaly more closely in most business cases. If the model outputs too many false positives, this can cost a company a large amount of human labour, and thus money [88, 58]. Training the model with the right data is one of the ingredients to ensure this.

The third challenge is the evolution of anomaly behaviour over time. This phenomena is also known as concept drift. Anomaly behaviour can for example shift due to a change in the process or because the users adapt their behaviour to the current detection models. The concept drift makes that anomaly generation will never be a finished process [23].

The last challenge is the presence of a selection bias. There will be a selection bias in what kind of anomalies are generated. This means that whoever is responsible for the generation process must be very careful about what is selected as the generating material and do the selection process very exhaustively.

1.4 Research question

To answer the described research gap, the following research question is proposed.

Main research question

How can anomalous sequences be generated to help train anomaly detection models to improve their performance?

The main research question is answered with the help of the following sub research questions.

1. What fraud characteristics are described in the scientific literature?
2. How can fraud characteristics be used to generated anomalous sessions?
3. To what extent would the generation and injection of anomaly behaviour in an event log help to improve the detection performance?
   (a) What is the effect of a higher percentage of anomalies in the data on the model performance?
   (b) What kind of anomalies can be detected well by the algorithms and which not?

1.5 Thesis outline

The remainder of this paper is organized as follows. First, in chapter two, we present the results from the background literature study, and in chapter three, the taxonomy of anomaly characteristics is given. Next, the research approach is presented in chapter four, followed by the approach in chapter five, the case study in chapter six and the results in chapter seven. Finally, the report is finished with the discussion in chapter 8 and the conclusion in chapter nine.

1.6 Partner organization

This thesis is conducted in collaboration with a partner organization, Capgemini Invent. Capgemini Invent is the management consultant brand of Capgemini[1] and helps other companies innovate and transform their business. Capgemini Invent provided the opportunity that started this research and is responsible for the practical support for this thesis. The scientific support is in the hands of Utrecht University.

[1]https://www.capgemini.com/service/invent/

2 Related work / background literature

The section below describes the background literature. This background literature aims to provide all the necessary background information that is needed to understand the topic of anomaly generation fully.

2.1 Process mining

Process mining is used to discover, monitor and improve actual processes [4]. The objective of process mining is to turn event data into insights and actions [68]. Process mining works with the process as it is and not how it is assumed to be. The process mining techniques do this by extracting information from event logs available in information systems. An event log is a made out of data that consist of three fundamentals. These are: timestamp, activity and case identifier. The timestamp records at which time the activity is executed. The activity records what process step was executed. The case identifier refers to the business case the activity belongs. An event log consist out of multiple cases, and each case can contain multiple events. In addition, other attributes can be stored. An example of an additional attribute that can be stored is the name or role of a person performing the activity.

2.1.1 Main tasks of process mining

There are three main tasks for which process mining can be used. These are: conformance checking, process discovery, and process enhancement [6]. Conformance checking uses a process model; a process model is a design of how the process should be executed. The purpose of conformance checking is to verify that the process has been performed as prescribed by the process model. The result of conformance checking is a list with events marked as conform or non-conform. Non-conform can either show that the activity is executed but does not occur in the process model or that the activity appears in the process model but not in the event log. A helpful example of how conformance checking can be used is for anomaly detection.

Another significant task of process mining is process discovery [3]. Process discovery techniques automatically construct a process model based on the given event log. For this process, they do not use any prior knowledge. The strength of process discovery is that it creates a model based on how it is executed. The results of the process discovery techniques can be compared to the process model of how the process is intended. This comparison can be used to discuss with all the stakeholders of the process on how the differences between the two models have occurred and how to fix the deviations.

The remaining main task of process mining is process enhancement [3]. Process enhancement is about extending or improving a current process model. The enhancement focuses primarily on identifying bottlenecks in the process model. This bottleneck detection can be done by calculating the time difference between the execution of all the activities. In addition, the extra stored variables can also be used to enhance the process. For example, by looking at the roles of persons who execute tasks that take too long, one can identify understaffed departments.

2.2 Anomaly detection

As explained earlier, an anomaly is a data instance that does not match the expected behaviour [37, 23, 62]. There are different terms that all refer to the concept of anomaly detection, and they are used interchangeably. The most used terms are deviations, outliers, novelties, and anomalies [79, 23]. The term anomaly was introduced in 1969 by Grubbs in his research [40]. Grubbs describes an anomaly as an instance that deviate considerably from other instances of the sample in which it appears. At that time, anomaly detection was mainly used for data cleaning since the appearance of anomalies in the dataset had a negative influence on the performance of the models used at that time [37]. Later on, the focus shifted towards detecting anomalies since the anomalies themselves are interesting and relevant for the domain in which they are detected. Goldstein and Uchida [37] add an extra condition that needs to be met by data instances to be regarded as an anomaly. This extra condition is that anomalies rarely occur in a dataset compared to the occurrence of normal instances.

Having discussed what an anomaly is, the following sections address the different techniques to detect anomalies.

2.3 Anomaly detection models

Many models can be used for the detection of anomalies. As said in the introduction, these models can be categorized into supervised, unsupervised, and semi-supervised learning. Below is an explanation of the models used in this research—these models were chosen since they are fast to use and easy to understand. For anomaly detection, it is essential that an explanation can be derived from the model on why a data instance was classified as an anomaly or not. These three models are suited for that task.

2.3.1 Logistic regression

Logistic regression is a type of model for probabilistic statistical classification. It can be used to make binary categorical values or multiclass predictions. It can predict the outcome from an infinite number of variables with an infinite number of values and returns a value between 0 and 1. The value it returns can be seen as a probability that a data instance belongs to a specific class [47].

Logistic regression uses a Sigmoid function to predict the outcome and a decision boundary to give the final label to the predicted data instance. Using the Sigmoid function and a decision boundary, individual data points less influence the logistic regression model. The Sigmoid function looks like an S-curve when plotted in a graph.

2.3.2 Decision tree

Decision trees are commonly used models for classification tasks. Essentially they learn a hierarchy relationship with if and else questions that lead to a decision [66]. The Decision tree starts with searching for the most informative split. Then, it looks for the purest split to determine the most informative split. This is calculated by an impurity metric, for example, the Gini-impurity. This impurity metric bases the split on the probability that an instance is misclassified due to the split and thus needs more following splits to be classified correctly.

Decision trees can be pruned until every node only contains values of the same class. A node containing only one instance or instances of the same class is named a leaf node. If a tree is fully grown, it consists of only leaf nodes. These trees and Decision trees, in general, are likely to overfit the training data and have a low ability to generalize to other datasets.

Decision trees are complex models since they have many hyperparameters influencing the model's performance. Examples of hyperparameters that a Decision tree can use are max depth, a minimum number of instances in each node, a max number of leaf nodes, a minimum impurity decrease and a max amount of features that can be considered for each split.

An advantage of a Decision tree is that it can be used to determine the feature's importance. The feature used to make the most informative splits can be classified as an essential feature. Next to this advantage, the results of a Decision tree are easy to understand, and the algorithm run time is not highly affected by an increasing data size since the number of considered splits does not depend on the size of the dataset but mainly on the number of features in a dataset.

2.3.3 Isolation forest

Isolation forest is a model-based approach developed in 2008 by Liu, Ting, and Zhi-Hua Zhou[64]. The approach used by an Isolation forest is fundamentally different from the standard. Instead of building a profile of normal behaviour and identifying which data instances do not conform to this normal behaviour profile, it attempts to isolate the anomalies from the rest of the data set.

Isolation forest is based on the following two characteristics of anomalies: Anomalies consist of fewer instances in the dataset, and anomalies have attribute values that contrast with normal instances. These two characteristics make anomalies more prone to isolation than regular points.

Isolation forest is built with decision trees. Each decision tree modelled by the Isolation forest is built as deep as necessary until all instances are a leaf node and thus isolated. The partitioning of each split is done randomly and the Isolation forest builds multiple trees. Isolation forest ensembles all the decision trees it builds together. Based on this forest of decision trees, the model calculates the average path length required to isolate the instance for all data instances.

Anomalies are isolated closer to the root of the tree and have a lower average path length, whilst normal points are isolated at the deeper end of the tree and have a higher average path length. The shorter the path to isolate an instance, the more likely it is that the data instance is an anomaly. This principle forms the method that is used by Isolation forests.

The result of the Isolation forest is a ranking that shows to what extent all the instances are anomalous. Isolation forest achieves the best results when the sampling size is kept small. Sub-sampling can contribute to a better result when working with high sample size. Isolation forest can also achieve good results in a high dimensional anomaly detection task.

2.4 Generation

There are numerous ways to generate data described in the literature. Below we discuss the overarching themes and for each method discuss there advantages and limitations.

2.4.1 Classical approaches

The most used classical approaches are Gaussian distribution manipulation, noise addition, rotation, scaling, warping, oversampling [61] and permutation [89]. These approaches are straightforward to implement. Gaussian distribution manipulation, for example, a change in standard deviation. New instances are generated and labelled as anomalies based on this changed standard deviation. Lin and Nadjm-Tehrani uses this manipulation of Gaussian distribution to create new anomalies. Alizadeh et al. [11] generate anomalies by adding noise to the dataset. The noise is generated by adding or removing some events in sessions. One advantage of the Gaussian distribution manipulation is that it can be easily performed. On the other hand, two disadvantages of Gaussian distribution manipulation are: the generated anomalies are rather distinct from normal instances, which makes them easy to detect by anomaly detection models, and the quality of the generated anomalies depends significantly on the available data [74].

2.4.2 Simulation-based models

Simulation-based models is about building a model that mimics a real system's behaviour based on explicitly defined physical laws and heuristic rules [32, 100]. When the simulator approximates real-world systems, they produce faithful data [74]. However, it is difficult for a simulation model to get so close to the real world, as it is challenging to tune the model's parameters to the correct values. In addition, running and tuning the model requires a high load of computational power. The last downside of simulation-based models is that they are not scalable to other problems.

2.4.3 Markov Models

Markov Models or Markov chains are a stochastic models that describes a sequence of possible events, where the probability of each event depends only on the state achieved in the previous event [34]. The dependence on the probability is called the Markov property. Markov models are widely adopted and easy to use [74]. The limitation of Markov models is that they have difficulty representing nonlinear relationships and their success depends on the amount of data available. An example of a visualisation of a Markov chain can be seen in Figure 1.

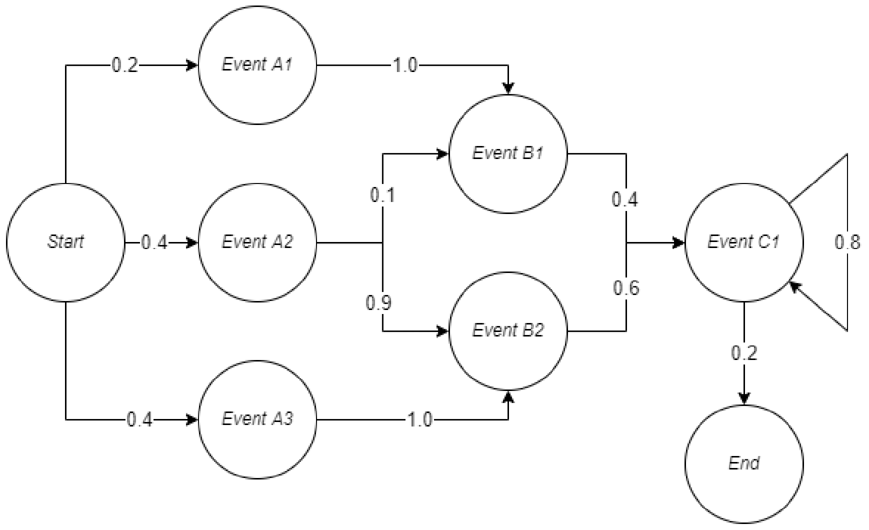


Figure 1: Example of a Markov chain

2.4.4 Generative Adversarial Networks

Generative Adversarial Networks (GAN) is a technique for augmenting datasets that can be used for both semi-supervised and unsupervised learning [26, 38]. GANs have their origins in data generation in image and video classification tasks and are currently applied in more domains. Estaban et al. [31] show in their research that GANs can be used for time-series modeling.

The GANs consist of two models: the generator model and the discriminator model. The discriminator is presented with both real and generated data instances and must classify them. The two models are trained together in a zero-sum game. Training continues until the discriminator is fooled half the time. The goal of training in this particular way is to generate data instances that are as close as possible to the actual data which makes them indistinguishable from real data [74].

The advantage of GANs is that it is suited for handling non-linear relationships. Due to their training method with the generator and discriminator function, they can generate data instances almost indistinguishable from real instances. A downside is that they are less suited for generating anomalies because of their training method.

2.4.5 Generation approaches summary

In total, we have discussed four approaches to generating data. The classical approaches are easy to use, but the results are generally of low quality and may lead to under- or overfitting. The simulation-based methods can produce good results, but it is not easy to approach reality with these models. Generating with Markov models is reliable but requires much data to capture all behavioural relationships appropriately. The last method we have explained is the GAN. This method can produce good results but is less suitable for generating anomalies. Given the advantages and limitations of all the approaches discussed in this chapter, we will continue the rest of the research with the Markov models.

3 Literature review of anomaly characteristics

The literature review of anomaly characteristics aims to establish a taxonomy of known anomalous behaviour and answers the first sub research question. The taxonomy includes several research domains to make it as comprehensive as possible. The snowball method was applied to the key articles of this literature research. The key articles are: Tax et al. [88], Goldstein and Uchida [37], Wang et al. [93], Xi et al. [97], and Van der Aalst [2]. These articles were snowballed—both backward and forward-looking. Backward-looking is looking at the references in the article. Forward-looking is looking at articles that refer to the article in question [96].

Next, in the search for relevant literature, the following search terms were used in Google Scholar: Process mining deviations, Process mining generate deviations, Anomaly detection, Anomaly generation, Credit card fraud, Insurance fraud, Process mining rework, Fraud characteristics.

A selection of relevant literature was made from these sources based on the criteria that the source was written in English, digitally available with the *University Utrecht Library Access*, and contained relevant information for the taxonomy. Furthermore, if possible, recent works were preferred over older ones. This chapter gives an overview of all the characteristics found in the literature. In the end, a table is presented with all the process mining characteristics to show the relevance of each characteristic individually.

3.1 Process mining anomaly characteristics

Process mining is the research field for discovering, monitoring, and improving processes as they are [1]. It is mostly about optimizing a current process. For an optimized process, all anomalies have to be removed. This can be done with conformance checking. For conformance checking, the event log is compared to the process model. If a case deviates too much from the process model, the case can be flagged as an anomaly. This makes process mining very suited for anomaly detection [62].

Process mining deviation characteristics are widely reported in the scientific literature. Below is a selection of relevant anomaly characteristics explained. The reported characteristics that overlap are merged.

The characteristics are divided into the following two categories: High-level and low-level process mining anomaly characteristics. This is done because the high-level characteristics do not translate in one pre-defined way to an event log, whereas the low-level characteristics translate in a straightforward way to an event log. We start by discussing the low-level characteristics since they are more relevant to the research.

3.1.1 Low-level process mining anomaly characteristics

Below the low-level process mining anomaly characteristics are discussed. To make these characteristics easier to understand, examples are presented. These examples show how the characteristics are translated to an event log. Where possible, the examples are explained by using the following event log as the ideal process model [51, 45, 84, 85, 18, 68, 67, 14, 80, 95, 69, 77, 46, 70, 54].

<Start_application, Input_info, Send_application, Receive_feedback, Accept_offer>

Skipped event or missing sequence. An event that should have been performed did not occur in the event log.
Example log: *<Start_application, Send_application, Receive_feedback, Accept insurance*. Input_info is missing.

Inserted event. An event that should not occur according to the process model is performed in the process.
Example log: *<Start_application, Check_conditions, Input_info, Send_application, Receive_feedback, Accept_offer>*. Check_conditions is added.

Loop on an event. An event is repeated while only a single event should have been performed. The repeated events take place one after the other.
Example log: *<Start_application, Input_info, Send_application, Send_application,, Receive_feedback, Accept_offer>*. Send_application is repeated.

Repetition of an event. An event is repeated while only a single event should have been performed. The event can be repeated later in the trace and does not have to follow each other immediately.
Example log: *<Start_application, Input_info, Send_application, Receive_feedback, Send_application, Accept_offer>*. Receive_feedback is repeated.

Swapping two events. Two events that must follow each other in strict sequence have changed their order.
Example log: *<Start_application, Send_application, Input_info, Receive_feedback, Accept_offer>*. Send_application and Input_info are swapped.

Replacing one event by another. An event takes place instead of another missing event.
Example log: *<Start_application, Receive_feedback, Send_application, Receive_feedback, Accept_offer>*. Input_info is replaced by Receive_feedback

Wrong roles / wrong resources. An unauthorized user executes an event.

Wrong attributes. The event or sequence contains attributes that do not correspond to the business process.

Wrong throughput time. The throughput time of a sequence of events does not match the time prescribed in the process model for that sequence. This can be too fast or too slow. The throughput time applies not only to sequences but also to individual events. An individual activity can also be performed too fast or too slow.

Wrong duty. A user performs two or more different activities in one running process.

Wrong decision. When a decision is made that does not conform to the process model. For example, there are two options for a loan application at a certain point, namely, accept or deny the loan. Then, however, a third option is executed, namely, send back to loan department.

Unexpected events. Events that are not expected at a certain point in the process but are still executed at that place. The term unexpected is highly dependent on the context.

Collective event anomalies. Events that are each independent inconspicuous, but their combination is suspicious. It is also highly dependent on the context.

Parallel. Activities that cannot be performed in parallel are performed simultaneously. This can be seen by checking the timestamp of the activities.

3.1.2 High-level process mining anomaly characteristics

There are two high-level process mining anomaly characteristics: Rework and workarounds. Both characteristics are high-level since it is not directly clear how the characteristics translate to the event log. We explain both characteristics in more detail.

Workarounds. A workaround is an intentional deviation, adaptation, improvisation, or other change from the designed path [13, 17]. The goal of a workaround is to reach the same result as the designed path but in an alternative way. A workaround is applied because a user experience hindrance in the designed path and wants to overcome this.

Workarounds can be viewed as either positive or negative [21]. A workaround is positive when the result of the workaround is of greater importance than the step that was skipped. For example, if a patient enters the hospital in a life-threatening condition, it may be a positive workaround to skip health insurance enrollment and begin medical treatment immediately. Conversely, a workaround can be seen as unfavourable when the appropriate step in the process is skipped too quickly, and the negative consequences are underestimated. In health care, this may be the case when a prescription is given without the consent of a physician with proper authorization.

The main idea of detecting workarounds is to map the barriers that make the workaround occur and see if the barrier can be remedied. There is, however, a difference between the goal of detecting a positive workaround and a negative workaround. A positive workaround is something that an organization can learn from and choose to make an established practice [42]. When a workaround becomes established practice, it is no longer considered a workaround. Research by van de Weerd et al. [94] shows concrete examples of helpful workarounds in the retail domain. Whereas a negative workaround is something that, if it goes unnoticed, can have negative consequences, such as loss of control, compliance issues, or lower process quality [19, 28, 60].

Workarounds are usually detected in a qualitative manner. Qualitative data on workarounds can be collected through interviews or observation. A disadvantage of both methods is that they are time-consuming and not scalable. Quantitative detection of workarounds can be done using process mining and, more specifically, the conformance checking technique of process mining. However, the quantitative part is best complemented by qualitative research [16].

Rework. A *rework* is a more specific type of process mining anomaly. Rework is the unnecessary repetition of work that results in wasted time and effort [78]. Reworks usually occur because the work done before was not performed correctly, and parts must be redone [14, 20]. Rework takes place after the work is done and usually there are a number of events between the original work and the rework. This is an important characteristic of reworks. Reworks are typically detected to detect and solve bottlenecks in a process, but can be used as anomaly detection characteristics in specific domains. For example, multiple reworks from a user performing a loan request can be marked as suspicious. Another possible rework is a sudden change in personal information when applying for an insurance product. In this example, a user reworks the personal information section. This rework can be done because a person made a mistake or wants to pay a lower premium by changing the zip code. With the help of domain experts, the suspicious rework sequences can be determined for each process.

3.2 E-commerce anomaly characteristics

Anomaly detection plays an important role in the E-commerce sector [88]. Its application has multiple purposes ranging from detecting fake reviews to detecting payments with stolen credit cards. Below is a selection of relevant characteristics presented and explained. The characteristics focus on the behaviour and actions that indicate anomalous activity. In the table at the end, more rule basic characteristics can be found. [63, 102, 105, 87, 93, 101]

Sensitive account changes. The change of sensitive information of an account at crucial moments in the process is a potential indication of anomaly behaviour. The connection between the change at the moment in the process is important. Every user can change his or her information from time to time. However, it should ring some bells when this happens at the checkout page. Password and mail address changes are the changes that happen the most in anomaly behaviour.

Device-sharing. If one device is used for multiple cases, there is a higher chance that these cases are anomalies. This is also the case for session sharing. A characteristic of anomaly behaviour is that multiple cases are performed in one session, e.g., buying two separate orders at the same webshop.

Synchronized timestamps. Synchronized timestamps mean that two different sessions have almost identical timestamps for the actions that they execute.

Random behaviour. Random behaviour can occur in two different types. The first is that a user immediately goes to the product that he wants to buy and right after goes to the checkout and buys the product. The other option is that a user starts to browse random pages and products after the target product has been put in the shopping basket.

Target item. Anomaly users visit the target item much more frequently than other users do. Next, they are more likely to add the target item to their favourite product list. Lastly, dishonest users are more likely to use the chatting functionality of a website before buying the target item.

Anomaly review. Anomaly users tend to review the target item with a severe bias and give the product much higher scores than other users do.

3.3 Credit card anomaly characteristics

User behaviour is commonly used in credit card fraud detection. Here they make a profile based on the historical transactions of a user. The incoming transactions are compared with the user profile, and if it deviates too much, a trigger is pulled [22, 24]. This is how credit card companies mostly do it, and for this method, they lean heavily on categorical variables or rule-based methods. For this research, we have looked at characteristics that indicate anomaly behaviour without having the historical data of a user. [30, 106, 52, 103, 104, 98, 97, 22, 8]

Change of PIN code or contract. The change of the PIN code of the credit card or an adjustment to the contract between the company and the user moments before the credit card is used for purchase can be an indication of anomaly behaviour.

Replacement or change request. If a replacement or change request is made close before or after purchase, this can indicate anomaly behaviour.

Frequency. The frequency of the use of credit cards can indicate anomaly behaviour. For example, it is anomaly behaviour if a credit card is used very often in a small amount of time.

Invalid number of login attempts. There is a higher chance of anomaly behaviour if there are many invalid login attempts before a transaction.

IP address and sessions. Payments from one IP address or in one session with multiple credit cards indicate anomaly behaviour. The other way around is also possible. Payments with one credit card from different IP addresses and sessions in a short period indicate anomaly behaviour. This characteristic is a good indication of payments with stolen credit cards.

Time of the day and week. The time of the day at which the transaction occurred can be an essential indicator of anomaly behaviour. For example, during the night, or when most people should be at work are such times. Also, the type of day can be an essential feature. For example, if it is a holiday, there is a higher chance of anomaly behaviour than on regular weekends. One can imagine that a transaction during Christmas Eve is more suspicious than a transaction during a typical Friday. The time and day-type features are powerful in combination with other features.

3.4 Bot anomaly characteristics

Percentages giving the share of bot traffic on the internet differ a lot. According to a report by Imperva, more than 40% of all the traffic on the internet comes from bots[2]. Not all bots are bad. Some help persons to find information quicker or get the cheapest plane tickets. However, on the other side are bots that help to spread misinformation about elections or

[2]https://www.imperva.com/blog/bad-bot-report-2021-the-pandemic-of-the-internet/

help with advertising fraud[3] [33]. These bots must be detected to prevent further damage. There are different methods and features used to detect these bots. The characteristics that affect the behaviour of the bots are presented below. [49, 33, 92, 90, 91]

High error accessing pages. Bots tend to visit more pages that do not exist than normal users.

Low page revisit. The behaviour of bots is such that they are less likely to return to a page they have previously visited. Bots are also visiting a lower number of distinct pages.

Suspicious sequence. There are possible combinations of actions that form a suspicious sequence that indicates that the user is a bot. An example is changing passwords and e-mail addresses in a short amount of time. Another example is visiting the pages in alphabetical order. Bots also have a smaller range of activities that they perform. They usually follow a strict set of behaviours. So, if a user shows behaviour that is always the same and in the same order, it is a good indication of anomaly behaviour.

Ignored cookie. If a user on a website ignores accepting or denying a cookie, there is an indication of anomaly behaviour and thus that the user is a bot.

Action time interval. The time between actions and page visits is faster for bots than for humans. The action time interval is thus a good indication of anomaly behaviour. The time offset, when a user does nothing between tasks, is also a good indication of anomaly behaviour.

Session actions. Bots have far fewer sessions than normal users do. Also, the duration of a session is shorter than that of ordinary users. Next to this is that the number of clicks of a bot in a session is lower, and the time between clicks is higher.

Active time. The behaviour when bots are active is different from when people are active. First, bots tend to be more active in the early morning hours. Next, bots are also more active during weekdays. This is because humans control bots, and these humans also have a regular work schedule.

[3]https://nos.nl/nieuwsuur/artikel/2415110-tientallen-miljarden-gaan-in-rook-op-door-online-advertentiefraude

3.5 Remaining anomaly characteristics

The remaining anomaly characteristics are found in the banking industry, fake reviews, medical claims, insurance fraud and spam detection. These domains are included to get the most comprehensive overview possible and show similarities between all the anomaly characteristics. [59, 10, 35, 56, 53, 25]

Device and session sharing. If multiple accounts are accessed from one device or in one session, there is a good indication of anomaly behaviour. This can be that one anomalous user accesses many different accounts or that many accounts are involved in transactions with small amounts of money.

Password failures. An increased number of password failures before an action takes place.

Change of zip code. Anomaly users are less likely to give a correct zip code. This is reflected in their behaviour by seeing events corresponding to changing the zip code.

Illogical sequence. An illogical sequence of events means that abnormal behaviour is detected when certain events follow one another. This should be based on domain knowledge. An example from the medical domain is a medical claim being filed before a patient arrives at the hospital. So what an illogical sequence is, is very domain-specific.

Time difference between actions. The time difference between actions is a good indication of anomaly behaviour. For example, two claims in a short period.

Active time. Potential anomaly behaviour is an activity of an account in the early morning hours when normally everyone is asleep. This can be an indication of using stolen information. Anomaly users can do this hoping that the legitimate owners are not awake and thus will not notice the use immediately.

For all the characteristics described in the background literature section, one should note that it always depends on the context if an observed event is an anomalous [18]. Next, in most cases, we can also develop examples that counter the characteristics. For example, in the case of password failure, it can always be that someone forgets his passwords and has to try multiple times before he guesses it correctly. In such a case, there are no wrong intentions with the actions. This shows that further research by a human is always necessary.

	Sensitive account changes	Device sharing	Synchronized timestamps	Random behaviour	Target item	Anomaly review	Change of PIN code or contract	Frequency	Invalid number of login attempts	IP address and sessions	Time of the day and week	High error accessing pages	Low page revisit	Suspicious sequence	Ignored cookie	Action time interval	Session actions	Active time	Device and session sharing	Password failures	Change of zip code	Illogical sequence	Time difference between actions	Active time
Skipped event				x	x								x	x	x							x		
Inserted event	x			x			x		x			x		x								x		
Loop on event				x				x	x			x		x						x	x	x		
Repetition of an event				x				x	x			x		x						x	x	x		
Swapping two events				x										x								x		
Replacing one event by another				x									x	x	x							x		
Wrong roles / wrong resources										x	x													
Wrong attributes		x	x							x	x						x		x					
Wrong throughput time			x	x				x			x					x	x	x				x	x	x
Wrong duty																								
Wrong decision																								
Unexpected events	x						x													x				x
Collective event anomalies													x											
Rework									x												x	x		
Workarounds													x									x		
Parallel								x						x	x	x						x		x

Figure 2: Linking the anomaly characteristics of the process mining domain to the specific characteristics found in the other domains

4 Research method

The research method chapter describes how this research has been conducted. First, the methodology that is used to shape the experiment is discussed. The chapter continues by describing the rationale and the design of the experiment. Finally, we conclude this chapter with an analysis of the validity threats.

4.1 Design science

We follow the Design Science Research Methodology by Peffers et al. [72]. Peffers methodology provides structured steps for designing an artefact and validating it by showing its competencies when applied to a problem in its context. The methodology of Peffers et al. consists of six steps, which are explained in the next paragraph.

The first step is defining the research problem and the relevance of a potential solution. In the second activity, the objectives of a solution are determined from the problem definition, and knowledge about the possibilities and feasibility of the research is gained. The third step is concerned with designing and building the artefact. The fourth step is followed by demonstrating the artefact in a use case. In the fifth step, the artefact is evaluated to see if it correctly solves the problems. The last step is communication, in which the artefact and the context in which it is applied are communicated to relevant stakeholders. Relevant stakeholders can be both researchers and practitioners. An overview of the methodology applied in this research can be seen in Figure 3.

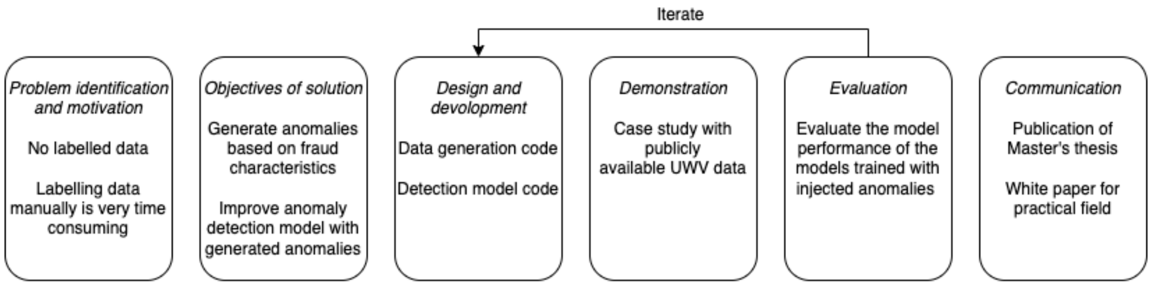

Iterate
Problem identification and motivation
No labelled data
Labelling data manually is very time consuming
Objectives of solution
Generate anomalies based on fraud characteristics
Improve anomaly detection model with generated anomalies
Design and devolopment
Data generation code
Detection model code
Demonstration
Case study with publicly available UWV data
Evaluation
Evaluate the model performance of the models trained with injected anomalies
Communication
Publication of Master's thesis
White paper for practical field

Figure 3: Research Methodology filled in for research conducted in this thesis

4.2 Case study: Rationale & Design

To answer the main research question, we start with a semi-structured interview and then conduct a case study. We opted for a semi-structured interview because it allows us to go deeper into what the interviewee had said and is open to discussing issues that were not anticipated when the interview questions were drafted. We selected a case study because of the flexibility in the research approach it provides and its ability to handle challenging and dynamic characteristics of real-world phenomena [96]. In addition, a case study allows the deployment of a designed artefact in its original context. Therefore, its impact and results are more realistic than controlled experiments and valuable for stakeholders and participants.

In total, we have tested three different anomaly patterns. For all three different patterns, the same case study setup is used. This is because the three patterns are very different and reflect most potential real-life anomaly cases. This means that by using three patterns, we can answer the research question with sufficient confidence. Furthermore, the same case study setup is used so that the results of the different patterns can easily be compared. An HP Elitebook with an Intel Core i5 @ 2.60Ghz processor and 16GB ram is used to execute this research.

4.3 Validity threats

We discuss the relevant validity threats for this experiment by Wohlin [96]. These relevant threats are conclusion, internal, construct and external validity.

4.3.1 Conclusion validity

Conclusion validity addresses the relation between the treatment and the results of an experiment. A threat to the conclusion validity is the use of the exploratory characteristics of this research. We counter this by describing all the steps as clearly as possible, using a public dataset, making the code available for this research public and using a correct experiment setup in which the treatment is the only thing being changed every time.

4.3.2 Internal validity

Internal validity refers to whether the measured effect is real or caused by external factors. A possible internal threat is that only one participant will be interviewed for this study. This person may be biased concerning the data, or their knowledge of the possible anomalies in the data may not be complete. The interview is conducted online and recorded. This can make the participants feel unable to speak freely and give desired answers. Since the participant is the only domain expert available, we must accept this potential threat and trust its expertise. A second threat is in the way we label the data. We do this with utmost precision and in consultation with the domain expert.

4.3.3 Construct validity

Construct validity asks if the experiment measured what is intended to be measured. We conducted a case study with three different patterns to make the experiment more robust. In this way, we ensure that a found result is not only based on chance on a single pattern. Another risk for the construct validity is that the interviewee may have misunderstood the questions as intended. We tried to counter this by contacting the interviewee in advance by e-mail so that he gets the questions on paper. We also try to be as neutral as possible during the interview and do not impose words on the participant. The last construct threat results from the program used by the UWV to record the behaviour of users on their website. This program may not capture all the behaviour or may be recording the wrong activity. Since the data has already been captured and captured by a third party, we cannot influence this threat and must accept it. By using all available data in the file, we try to mitigate this threat as best we can.

4.3.4 External validity

External validity concerns the research's ability to generalize to other contexts. A threat is that we use a case study; thus, we cannot draw a sample from a population to test a hypothesis statistically. However, we use a dataset from an actual situation for this research. Other organizations can quickly obtain the format of this dataset. This increases the generalizability of the research. Another positive aspect of the research is the use of the general patterns extracted from the taxonomy. These can also be easily applied in other contexts.

5 Approach

In the approach section, we explain our approach to anomaly generation. In section 5.1, we first describe how we convert an event log into a Markov model. Then, in section 5.2, we describe the three general patterns we have designed, and in the last subsection, we describe the practical implementation of the designed patterns in the Markov model.

5.1 Markov Model

A Markov model was used to model and generate anomalies. An example of a Markov model is shown in Figure 1. We use all sessions and activities available in the dataset. To explain the steps taken, we use an example log which can be seen in table 1.

Session ID	CustomerID	Activity	Timestamp
Session 1	1	Start_application	2015-11-06 08:07:22.780
Session 1	1	Input_info	2015-11-06 08:07:40.767
Session 1	1	Send_application	2015-11-06 08:07:51.390
Session 1	1	Accept_offer	2015-11-06 08:08:06.003
Session 2	101	Send_application	2016-02-28 08:17:15.947
Session 2	101	Accept_offer	2016-02-28 08:18:31.454
Session 3	224	Input_info	2016-01-14 08:32:11.511
Session 3	224	Send_application	2016-01-14 08:34:12.123
Session 3	224	Accept_offer	2016-01-14 08:37:23.984
Session 4	7653	Start_application	2016-02-20 20:15:10.321
Session 4	7653	Input_info	2016-02-20 20:16:09.647

Table 1: Example log for Markov model explanation

5.1.1 Counting pairs

To build the Markov Model, we use the SessionID, Timestamp, and the URL_FILE (activity) variable. After loading the dataset, it is sorted by the session identifier variable as the first variable and timestamp as the second variable. This is done to ensure that all activities are in the order they are performed by the users. Then, we group all activities by their SessionID. For the activities in each session, we match it to its consecutive activity. If the activity is the last in the session, the consecutive activity is a Not a Number (NaN) value. These NaN values are changed to the string ”end_session”. With the help of this string, we can easily calculate the probability that a session ends after a specific activity. The results of these actions are stored in a new variable in the DataFrame. This resulting DataFrame can be seen in Table 2.

Session ID	Activity	Consecutive activity
Session 1	Start_application	Input_info
	Input_info	Send_application
	Send_application	Accept_offer
	Accept_offer	End_session
Session 2	Send_application	Accept_offer
	Accept_offer	End_session
Session 3	Input_info	Send_application
	Send_application	Accept_offer
	Accept_offer	End_session
Session 4	Start_application	Input_info
	Input_info	End_session
Session 5	Accept_offer	End_session

Table 2: DataFrame after calculating the consecutive activity

When we have made all the pairs of activity and their consecutive activity, we count for each pair how often they occur and store the pair and the count in a DataFrame. The results from this action can be seen in Table 3.

Activity	Consecutive activity	Count
Accept_offer	End_session	4
Input_info	Send_application	2
Input_info	End_session	1
Send_application	Accept_offer	3
Start_application	Input_info	2

Table 3: DataFrame after counting pairs

Code implementation. For counting the pairs, we have used Python as the programming language. We start with sorting the DataFrame in the correct order with the .sort_values() functions[4]. In this function, we have specified that it first needs to sort

[4]`https://github.com/JochemVeldmanUU/Thesis_anomaly_generation/blob/main/Markov_model_probability.ipynb`

on the SessionID variable and second on the TIMESTAMP variable. For grouping the DataFrame, we used the .groupby() function, and for obtaining the consecutive activity, we used the .shift(-1) method on the groupby object. To turn the NaN values into a usable string we executed the .fillna("end_session"). For counting the pairs, we use the Counter() function. The result of the Counter() function is stored in a DataFrame. To get it in the correct format, we first transpose the DataFrame with .T and then reset the index with the reset_index() function.

5.1.2 Markov model table

The DataFrame containing the pairs and counts is tilted into a pivot table. The pivot function converts the activity to the index of a row, the successive activity converts to the index of a column, and the count becomes the value of the row and column. Pivoting results in combinations of activity and sequential activity that do not exist in the dataset and are presented as a NaN value in the returning DataFrame. These NaN values are converted to a 0 value because a 0 indicates no relationship between the activity and the successive activity. The table that show the result of this process can be seen in Table 4.

	Accept_offer	Input_info	Send_application	Start_application	End_session	Sum
Accept_offer	0	0	0	0	4	4
Input_info	0	0	2	0	1	3
Send_application	3	0	0	0	0	3
Start_application	0	2	0	0	0	2

Table 4: DataFrame resulting from pivot action with sum of all activities

The Markov model needs the probability from an activity to a consecutive activity. The pivot table returns absolute numbers in the form of the count of the pair. We transform these counts into probabilities by summing all the values in a row and then dividing all the counts by the sum of the row. This step can be seen in Table 5.

	Accept_offer	Input_info	Send_application	Start_application	End_session
Accept_offer	0	0	0	0	1
Input_info	0	0	0,67	0	0,33
Send_application	1	0	0	0	0
Start_application	0	1	0	0	0

Table 5: DataFrame resulting from pivot action probabilities

The final step in making the Markov model table is adding the probability that an activity is a session's start activity. This is done by taking the first activity of all sessions. We then count how many times each activity occurs and divide these counts by the total sum of all sessions. This results in the probability that an activity is the starting activity. The results of this final step can be seen in Table 6.

Starting activity	Frequency	Starting probability
Accept_offer	1	0,2
Input_info	1	0,2
Start_application	2	0,4
Send_application	1	0,2
Sum	5	1

Table 6: Counting frequency of activity being the start activity

These start probabilities are merged with the DataFrame resulting from the pivot function. Not all activities take place as a start activity. When the DataFrame is merged with the starting probabilities, these activities are not in the starting probability list and have a NaN value when the two are merged. These NaN values are converted to a 0 value because a 0 indicates no chance of the activity becoming a starting activity. The final resulting DataFrame can be seen in 7.

	Accept_offer	Input_info	Send_application	Start_application	End_session
Start_session	0,2	0,2	0,2	0,4	0
Accept_offer	0	0	0	0	1
Input_info	0	0	0,67	0	0,33
Send_application	1	0	0	0	0
Start_application	0	1	0	0	0

Table 7: Final DataFrame from the example event log with all probabilities needed to run a Markov Model

The final table contains a total of 587,766 values equal to zero. Out of a total of 603,713 occurrences, this is a considerable proportion. Working with this table is computationally expensive. We are converting the table into a dictionary because this results in a format that allows the removal of all zero instances from the final product. When we use the dictionary for the generation, we save much time compared to the DataFrame since we do not have to loop through all the redundant values.

Code implementation. To create the pivot table, we have used the Pandas' function .pivot_table(). We have implemented this function in a custom function make_pivot()[5]. In the pivot table we transform all the NaN values to 0 with the .fillna(0) function. The function .sum(axis=1) is used to get the sum of the total performed activities in a row. All the values in the pivot table are divided by this sum with the .div(axis=0) function. To get every first activity in a session, we groupby the DataFrame on SessionID and use .nth(0) to get the first row with the first performed activity. The resulting list with all first activities of all sessions is counted with the .value_counts() function. To get the starting probability of each activity we divide the individual values from the .value_counts() methods with the sum of the .value_counts(). We obtain this sum with the .sum() function.

5.2 Patterns

Based on the taxonomy of fraud characteristics, we have designed three general patterns. The idea behind these patterns is that they can be applied to all contexts where event data is available. For the design of the patterns, we have tried to design patterns that are as comprehensive as possible with the characteristics described in the taxonomy.

5.2.1 Repetition

The repetition event is often described in the literature as an anomaly characteristic in process mining. It consists of a single activity that can be performed more or less than usual. A semantic representation of the repetition pattern can be seen in Figure 4.

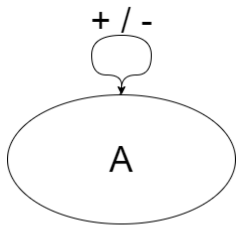


Figure 4: Overview of repetition pattern

5.2.2 Direct-follow

The direct-follow relation consists of two activities and one directed link between them. A directed link is a one-way link between the two activities. It is possible to go from A to

[5]https://github.com/JochemVeldmanUU/Thesis_anomaly_generation/blob/main/Markov_model_probability.ipynb

B, but it is impossible to go from B to A. In the direct-follow relation, we can manipulate the probability that the active state goes from activity A to activity B. The direct-follow relation results from multiple characteristics described in the taxonomy. The pattern results from the skipped event, inserted event, swapping two events and replacing one event with another characteristic. The direct-follow relation can be seen in Figure 5.

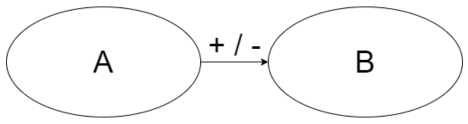


Figure 5: Overview of direct-follow pattern

5.2.3 Advanced repetition

The advanced repetition pattern consists of three different activities. Activity A, links to activity B and C. Activity B and C link back to activity A. All the links between the activities are direct links. This creates a loop between activity A and B and between activity A and C. This pattern results from the repetition characteristic in the taxonomy.

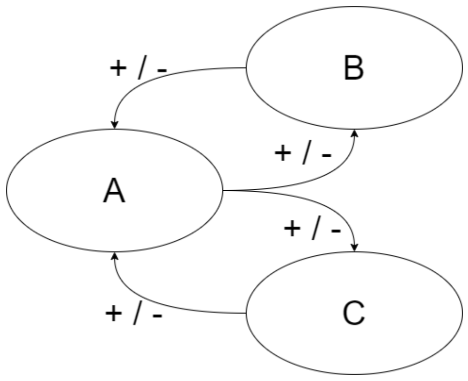


Figure 6: Overview of advanced repetition pattern

5.3 From patterns to Markov models

After creating the final DataFrame for the Markov model, we need to implement the patterns in the model to generate anomalous sessions. We do this by changing the *deviation rate* parameter. This parameter indicates by how much the considered probability in the Markov model is going to change. First, we select the correct data instance in the DataFrame that needs to be manipulated. This can be seen in Table 8. Then we overwrite this data instance with a new value. This new value is the old value plus or minus a number. In table 9, we added 10 to the old value of 0. After changing the probability, we need to sum all transition probabilities and divide all transition probabilities in a column by the sum of probabilities of that column. This recalculation must be done as the sum of the probabilities must equal one. An example of the result of this step can be seen in Table 10.

	Accept_offer	Input_info	Send_application	Start_application	End_session
Start_session	0,2	0,2	0,2	0,4	0
Accept_offer	0	0	0	0	1
Input_info	0	0	0,67	0	0,33
Send_application	1	0	0	0	0
Start_application	0	1	0	0	0

Table 8: Visualise which probability is going to change for the repetition pattern

	Accept_offer	Input_info	Send_application	Start_application	End_session	Sum transition probabilities
Start_session	0,2	0,2	0,2	0,4	0	1
Accept_offer	0	0	0	0	1	1
Input_info	0	0	0,67	0	0,33	1
Send_application	1	0	0 + 10	0	0	11
Start_application	0	1	0	0	0	1

Table 9: Markov model table with changed probability

	Accept_offer	Input_info	Send_application	Start_application	End_session
Start_session	0,2	0,2	0,2	0,4	0
Accept_offer	0	0	0	0	1
Input_info	0	0	0,67	0	0,33
Send_application	0,09	0	0,91	0	0
Start_application	0	1	0	0	0

Table 10: Changed probability in Markov model

5.4 Generate with Markov Model

For generating the anomalous sessions, we have created a custom function. This function generates the requested number of anomalous sessions. This parameter is given to the function science since we use the number of added anomalous sessions to the training data as our second parameter for the experiment. The generation of the session starts with determining the start activity. All the activities and the corresponding probabilities from the ”Start_session” row are taken, and a random activity is chosen based on the given probabilities. E.g. in the example presented in Table 7 this would mean that there is a 20% probability that ”Input_info” will be the start activity. Let us assume that this activity is indeed selected as the starting activity. The starting activity is appended to a temporary sequence, and the function then takes all the related activities and probabilities from the ”Input_info” activity. In this, the corresponding activities are ”Send_application” with a 67% probability and ”End_session” with a 33 % probability. Let us assume that the function has picked the ”End_session” activity as consecutive activity. This activity is added to the session, and all the activities that form a session are saved together.

Generate with Markov model implementation. For generating we have created the custom markov_chain() function[6]. This function takes in the number of sessions we want to generate, the start probabilities for the activities and the dictionary containing all the transition probabilities. The functions backbone is a nested while loop. The first while loop continues until the amount of anomalous sessions is generated. The second while loop stops when the ”End_session” activity is selected. For selecting the activities, we make use of the random.choice() function from Numpy. To save the activities and the sessions, we use a nested list. The activities are appended to a temporary list. If the ”End_session” activity is selected, the temporary list is appended to the main list, and the provisional list is emptied. After this, the process of generation starts again.

5.5 Chapter summary

In this chapter, we explained how we model the event log as a Markov model and apply our designed patterns to the Markov model to generate anomalous sessions. We also introduced two parameters that we can control in the experiment. The first is the *deviation rate*, and the second is the *number of injected anomalous sessions*. These two parameters are used to investigate the effect of injecting more generated sessions into a training set and to see the effect of making anomalies more anomalous. The following chapter will show how we have applied this approach to a case study.

[6]`https://github.com/JochemVeldmanUU/Thesis_anomaly_generation/blob/main/Markov_model_probability.ipynb`

6 Case study

In this section, we describe how we have conducted the case study. We explain how we translate the general patterns to specific patterns for the UWV dataset and inject the generated anomalies into the training data.

6.1 Data description

6.1.1 Data context

The case study is conducted with a public dataset from the Business Process Intelligence Challenge (BPIC) 2016[7]. The BPIC is an annual challenge in which an external organisation provides a real data set from the practice of their domain. The external organization in 2016 was: Uitvoeringsinstituut Werknemersverzekeringen (UWV). The UWV is a dutch governmental organization responsible for all the social benefits in the Netherlands.

In total, five files are provided by the UWV. The first file is clicked data from the websites where users are not logged in, and the second is click data with logged-in users. The third file contains questions asked by customers, the fourth contains messages sent by customers, and the last file contains complaints filed by customers. The UWV has anonymized all data. In this experiment, we only use the second file, the click data with logged-in clients. This file is used because it contains the most detailed behaviour per session. Each URL file clicked by a user is recorded as the activity performed. To our knowledge, the UWV data is the only publicly available dataset with recorded clicks on a website in a data format that is suited for process mining.

Users can use the website from the UWV for multiple purposes. This also means that no predetermined roadmap exists that all users follow. Users can use the website to upload their resumes and other files needed to apply for a job. They can search for vacancies on the website, and the UWV also gives them personalized job openings based on their resume. The final feature of the website is that users can apply for vacancies.

6.1.2 Data processing

The event data can be easily downloaded from the dedicated website. The file is already in a CSV format and only requires a little pre-processing to be analysed. First, the timestamp variable had to be transformed from a string to a timestamp format, and before loading the data, the separator had to be specified, as this was not the standard for a CSV file. Next, we excluded the rows that caused an error when loading the data. This was done since it only

[7] https://www.win.tue.nl/bpi/doku.php?id=2016:challenge

considered a negligible amount of rows compared to the amount of effort for including them. For loading and analysing the data, we used Python 3.9.7 and Disco[8]. For Python, we used Pandas, Numpy and Sci-kit learn as our main libraries. All the programming was done in Jupyter Notebook. We use Python because this programming language and its libraries are very suitable for processing and handling data. Disco is used because it is very light and the program with the most features when using a free student account.

6.1.3 Data overview

The UWV has provided a data dictionary that contains information for all the variables in the dataset[9]. The data provided cover the period from 6 November 2015 to 28 February 2016. The data consists of 7.174.934 rows and has a size of 1,06GB. A total of 781 unique activities have been performed in 660.270 sessions. The average number of activities per session is 10,87, and sessions come from 26.647 unique users. All the sessions originate from 61.401 unique IP addresses. An overview of all the variables can be found in Table 11. The variables that have our interest for this experiment are CustomerID, SessionID, TIMESTAMP and URL_FILE. The table shows that for these four variables there are no missing values.

6.1.4 Activity distribution

The distribution of the activity feature follows a power law distribution, see Figure 7[10]. Power law distribution means a small number of instances is responsible for most of the occurrences, and a large number of instances is accountable for a low amount of occurrences. For example, the three most performed activities make up 50,97% of the executed activities in this distribution. The twenty most performed activities make up 95,52%. When looking at the activities with a low amount of executions, 615 activities were performed less than 200 times, 357 activities were executed less than ten times, and 149 activities were conducted a single time. All other distributions discussed in this section follow the same power law distribution as seen in Figure 7. For this reason, we place all the visualisations of these distributions in Appendix C. In this Figure 7 we have plotted the activities on the x-axis and the occurrence of the activities in the dataset on the y-axis. The activity names are not plotted since all the names would not fit on the plot.

[8]https://fluxicon.com/disco

[9]`https://www.win.tue.nl/bpi/lib/exe/fetch.php?media=2016:data_dictionary_bpi_challenge_2016.pdf`

[10]`https://github.com/JochemVeldmanUU/Thesis_anomaly_generation/blob/main/Data%20visualization.ipynb`

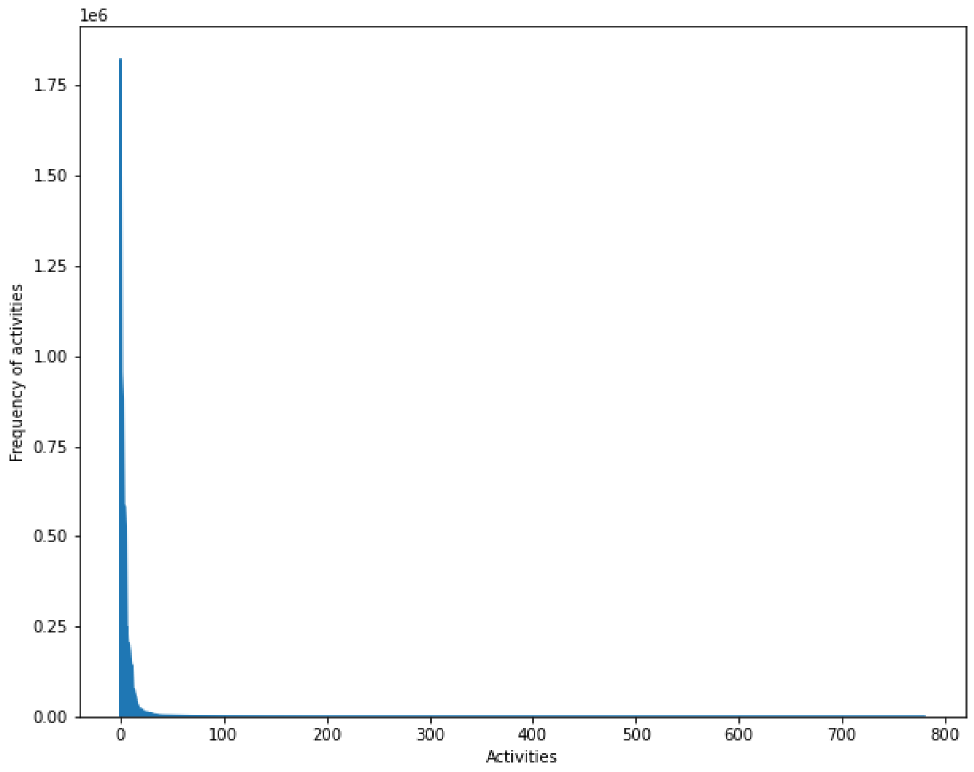


Figure 7: Distribution of activity frequency

Table 12 shows the ten activities that are most performed by the users. These most performed activities are all general activities; from the activity alone, it is hard to tell what a user is doing on the site. For this reason, we need to look at all activities in a session to analyse the data from the UWV.

6.1.5 Activities per session

There is a total of 660.270 sessions present in the dataset. Of these sessions, 53.410 sessions only include one activity. 228.512 sessions include less than five activities, and 453.274 sessions include fewer activities than the average of 10.8 per session. When looking at the most active users in the data, we see that the 1% most active users are responsible for 8,08% of all the executed activities on the website. In Figure 34 in Appendix C the dis-

Activity	Total frequency	Unique frequency
CustomerID	7.174.934	26.647
AgeCategory	7.174.934	4
Gender	7.174.934	2
Office_U	7.174.934	13
Office_W	7.174.934	40
SessionID	7.174.934	660.270
IPID	7.174.934	61.401
TIMESTAMP	7.174.934	7.164.782
VHOST	7.174.934	2
URL_FILE	7.174.934	781
PAGE_NAME	7.174.934	600
REF_URL_category	514.099	12
page_load_error	7.174.934	12
page_action_detail	11.150	1107
tip	5429	33
service_detail	223.051	18
xps_info	48.983	217

Table 11: Overview of variables present in the datafile with users logged in.

tribution of the activities per session is plotted. The SessionIDs are plotted on the x-axis, and the frequency of executed activities in a session is plotted on the y-axis. The SessionIs are not plotted since these would not fit on the plot.

The distribution of activities per session tells us that users follow many different patterns. Therefore, this distribution must be considered when applying the general patterns to the dataset because there can be patterns unsuitable for the generation experiment. Fortunately, there are 660,270 sessions, so that enough sessions can be found. This makes SessionID a suitable variable to group all activities on.

6.1.6 Activities per customer

When looking at the activities executed per customer, we find an average of 269,25 activities per user for the 26.647 unique users. One hundred twenty-three users have only executed a single activity, and 905 have performed ten or fewer activities. A total of 17.612 have performed less than the average number of activities per user. See Figure 35 in Appendix C for a visualisation of the distribution of activities per customer. On the x-

Activity	Frequency	Percentage of total	Cumulative
/werk_nl/werknemer/mijn_werkmap/doorgeven/taken	1822421	25.40	25.40
/werk_nl/werknemer/mijn_werkmap/werk-zoeken/vacatures_bij_mijn_cv	953968	13.30	38.70
/werk_nl/werknemer/mijn_werkmap/werk-zoeken/mijn_cv	880580	12.27	50.97
/werk_nl/werknemer/home	582726	8.12	59.09
/werk_nl/werknemer/mijn_werkmap/werk-zoeken/vacatures_zoeken	582624	8.12	67.21
/werk_nl/werknemer/mijn_werkmap/postvak/mijn_berichten	528030	7.36	74.57
/portal/page/portal/home/diensten/aanvragen-ww	251063	3.50	78.069
/werk_nl/werknemer/mijn_werkmap	206792	2.88	80.95
/werk_nl/werknemer/mijn_werkmap/doorgeven/mijn_sollicitaties	203828	2.84	83.79
/werk_nl/werknemer/werkmap	181207	2.53	86.32

Table 12: Overview of ten most performed activities

axis, we have plotted the customers, and on the y-axis, the number of activities they have performed. The CustomerIDs are not plotted since this would not fit on the plot.

The activities are unevenly distributed among all customers. Compared to the SessionID variables, there are fewer number of unique customers. This makes the CustomerID variables less suitable than the SessionID variable for grouping all activities.

6.1.7 Sessions per customer

Users perform an average of 24,79 sessions. Nine hundred forty-two users have only performed actions in a single session, and 8185 users have executed less than ten sessions. 17.170 make use of less than the average of performed sessions per user. The most active 1% of the users execute 6,86% of all the sessions in the dataset. See Figure 36 in Appendix C for a visualisation of the distribution. On the x-axis, we have plotted the customers, and on the y-axis, the number of sessions they have performed. The CustomerIDs are not plotted since this would not fit on the plot. Since most users perform a low number of sessions, grouping by CustomerID and SessionID is not a suitable option.

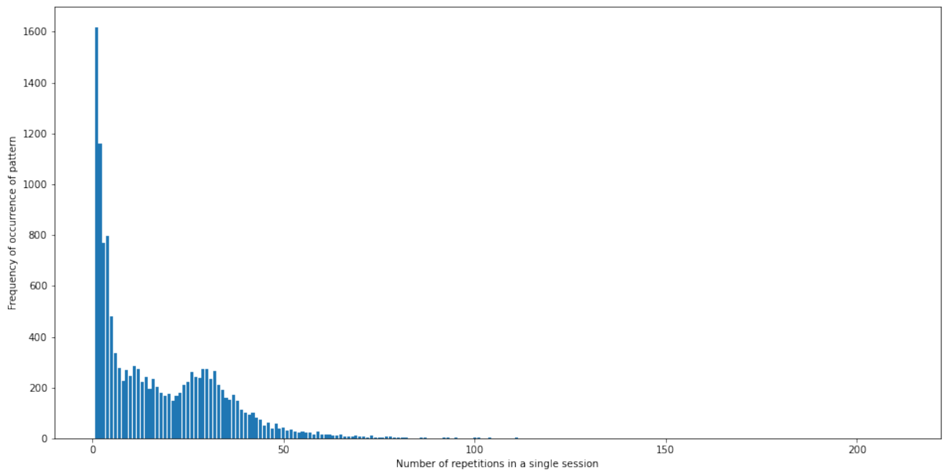


Figure 8: Distribution for repetition pattern

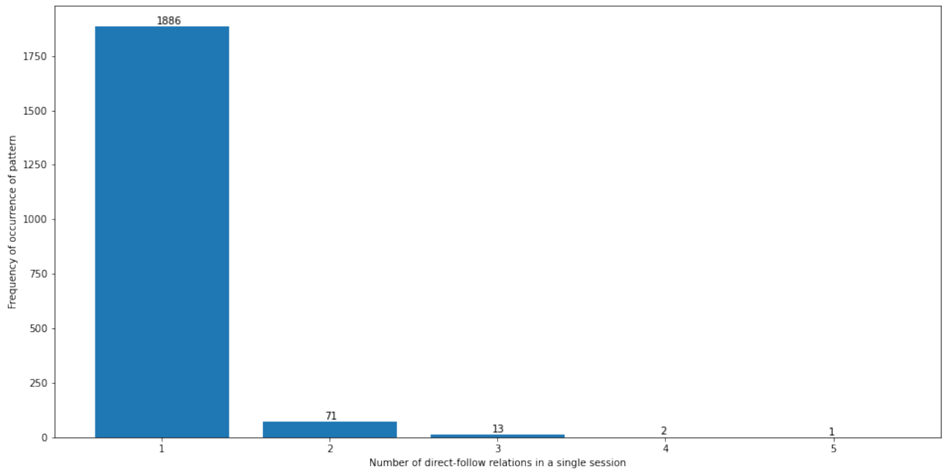


Figure 10: Distribution for direct-follow pattern

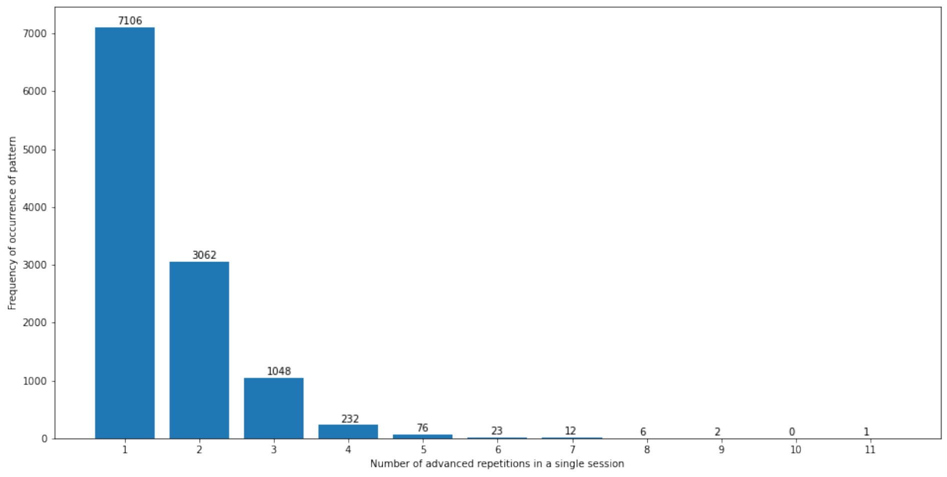


Figure 9: Distribution for advanced repetition pattern

6.1.8 Pattern distributions

To get a better understanding of the three patterns in the UWV data, we have visualised all three patterns. These distributions can be seen in Figure 8, 10, and 9. In the plots, we visualised the distributions of the patterns before injecting generated anomalies into the training data. The main idea behind showing these distributions is to get an overview of what they looked like before we injected anomalies and that we can compare them to the distributions when we injected anomalies.

6.2 Interviews

6.2.1 Beginning of interview

At the beginning of the case study, an interview was held with the UWV employee responsible for the dataset. The participant was approached by e-mail. In the first e-mail sent, we explained the reason and purpose of conducting the study. Along with this explanation, we sent the relevant features of the anomaly from the literature, asking whether it is possible to talk about the domain of the dataset and how the features can be applied to the domain.

After the participant's consent, we planned an online meeting. The meeting took place online because of the travel distance and was considered safe given the prevailing COVID-19 advice of the Dutch government. The interview was conducted in a semi-structured format. At the beginning of the interview, the participant was asked for permission to record the interview. This was done so that the interview could be listened to later if any information was unclear. During the interview, the participant provided more information about the domain and discussed how the characteristics of the anomaly apply to the domain. The correspondence with the UWV employee can be seen in Appendix B.

Domain	Anomaly characteristic
Process mining	Repetition of activity
	Wrong throughput time
	Loop on activity
	Wrong attributes
	Timestamps
E-commerce	Random behaviour
	Synchronized timestamps
Credit card	Ip address and session sharing
	Time of the day and week
	Invalid number of login attempts / Password failures
Bot anomaly	Low page revisit
	Action time interval / Time difference between actions
Remaining	Total active time
	Illogical sequence

Table 13: Overview of relevant characteristics according to the UWV domain expert

6.2.2 Case study: First interview

The purpose of the first interview is to gain more domain knowledge and to conclude which found characteristics from the scientific literature apply to the UWV website. In terms of the domain knowledge gained, we discovered that there is no single process underlying the UWV website. Users may visit the UWV website for multiple purposes, and during the months, the user's behaviour on the website may also change. Users usually start by uploading their resumes. After uploading their resume, they can search for eligible jobs. The final step in the process is to apply for the job. Users can go back and forth between these steps, and it is difficult to understand which higher-level task they are currently performing. The absence of an underlying process makes it more difficult to distinguish normal behaviour from abnormal behaviour. After gathering the domain knowledge, we discuss with the employee of the UWV what kind of behaviour is normal in the domain and what kind of behaviour could be classified as anomalous. Based on this more general information, we review every anomaly characteristic and examine which characteristics are involved in the domain. In this review, we distinguish process mining anomaly characteristics and anomaly characteristics from the specific domains. The relevant characteristics can be seen in Table 13. We also discussed with the domain expert how the three general patterns could be applied.

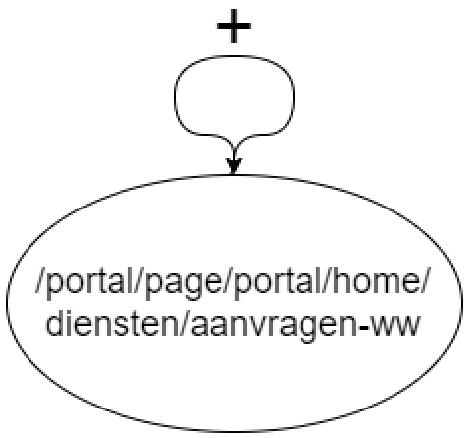


Figure 11: Overview of applied repetition pattern

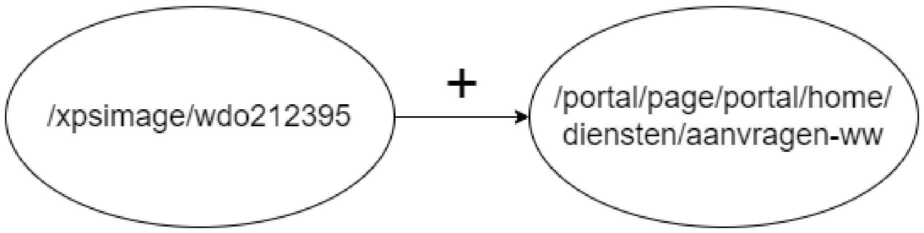


Figure 12: Overview of applied direct-follow pattern

6.2.3 Case study: Second interview

With the results from the first interview we have looked at the data to see in more detail to which parts of the domain the anomaly characteristics apply. For this we looked at how relevant an activity was based on the information we had of the activity and the occurrence of the activity in the data. Based on this requirements we where able to apply the repetition pattern to the domain. The applied repetition pattern can be seen in Figure 11. This pattern was the repetition of the activity ”aanvragen-ww”, which is requesting social benefits. After this analysis we had our second contact with the domain expert from the UWV. The domain expert confirmed that a repetition in this activity is of interest to the domain and can potentially be considered as anomalous behaviour. For the direct-follow and advanced repetition he also suggested activities to look into that where related to the ”aanvragen_ww” activity. Based on this information we where able to apply the other two pattern in the domain. The direct-follow pattern can be seen in Figure 12 and is a direct link from the ”/xpsimage/wdo212395” activity to the ”aanvragen_ww” activity. The xps image activity is an activity in which the user views an image or document in the browser. For the advanced repetition pattern we use ”aanvragen_ww”, ”home” and ”diensten/overzicht”. A visualisation of this applied pattern can be seen in Figure 13. The two applied patterns where again validated with the expert.

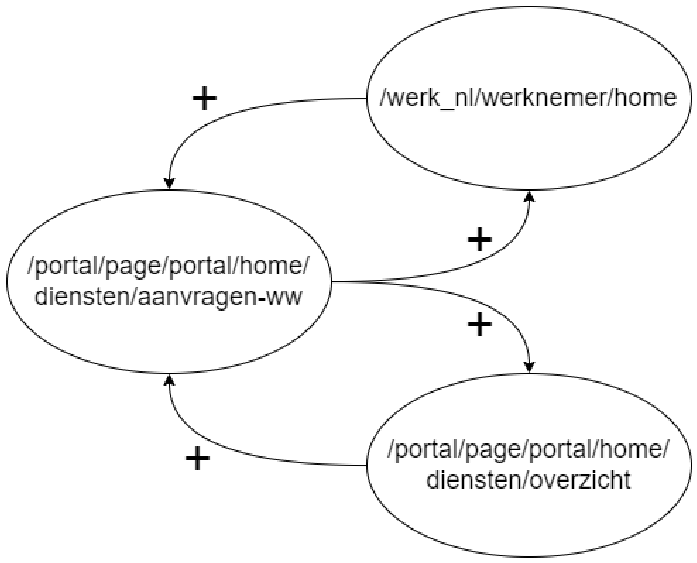

/werk_nl/werknemer/home
/portal/page/portal/home/diensten/aanvragen-ww
/portal/page/portal/home/diensten/overzicht
+
+
+
+

Figure 13: Overview of applied advanced repetition pattern

6.3 Experiment evaluation

6.3.1 Experiment setup

After pre-processing the data, we conduct the experiment. Figure 14 shows a schematic overview of the experiment. We split the data into a train and test set in the experiment.

The training data is enriched with generated anomalies. The model is trained on the train data, and the trained model predicts the label for all the sessions in the test set. These predictions on the test sets are used to compute the performance scores, which can be evaluated. Based on the evaluation, we conclude if adding the generated anomalies to the training data helps to detect anomalies more accurately. In the following, we discuss the training and testing of the model with the generated anomalies.

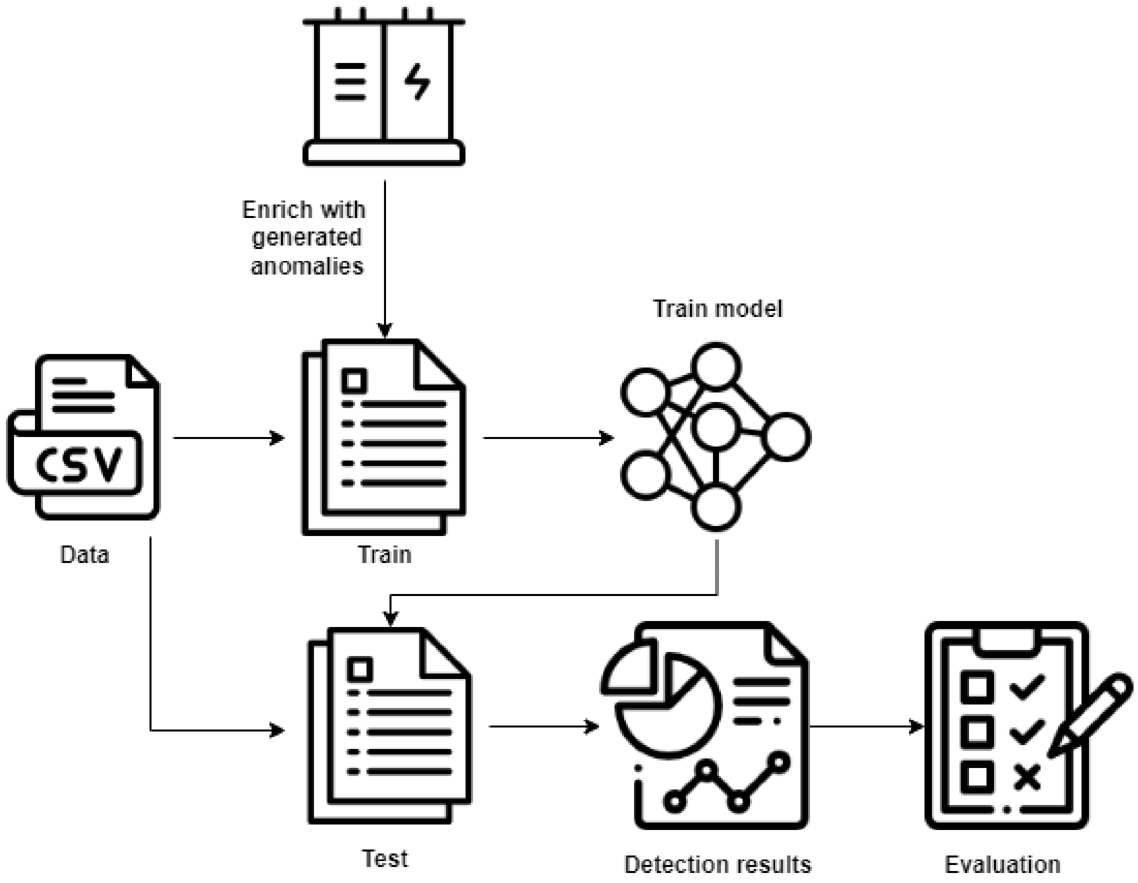


Figure 14: Overview of experiment setup

6.3.2 Case study: Labeling the anomaly patterns

To start the experiment, labelled data instances must be obtained. The original dataset contains no labelled data. Based on the taxonomy's characteristics and the domain expert's information, we have developed the following strategy to obtain labels. For the repetition pattern every session that includes more than fifty repetitions (>50) of the activity "/portal/page/portal/home/diensten/aanvragen-ww" is considered as an anomaly. With repetition, we mean consecutive execution of the same activity where there is no other URL visited in between. This labelling of the data results in 465 sessions that are marked as an anomaly. The other sessions in the data are marked as normal instances, and the total dataset is exported to a new CSV file [11].

For the direct-follow pattern, we label all sessions as an anomaly when it has more than one (>1) occurrence of a direct link from "/ xpsimage/wdo212395" to the activity "/portal/page/portal/home/diensten/aanvragen-ww". This results in 87 sessions that are labelled as anomalies. The rest of the sessions are labelled normal, and the total dataset is saved in a new CSV file[12].

For the advanced repetition pattern, we label each session as an anomaly with more than three (>3) non-consecutive executions of the activity "/portal/page/portal/home/ services/requests-ww". This results in a total of 352 labelled anomalies, and all the other sessions are classified as normal instances. The pandas DataFrame is saved in a new CSV file [13].

	Repetition pattern	Direct-follow pattern
Deviation rate = 0	0.928	0.159
Deviation rate = -0.1	None	0.066
Deviation rate = -0.55	0.841	None
Deviation rate = 0.2	0.940	0.299
Deviation rate = 1	0.964	0.580
Deviation rate = 3	0.982	0.790
Deviation rate = 5	0.988	0.860
Deviation rate = 10	0.993	0.924
Deviation rate = 25	0.997	0.968

Table 14: Manipulation of probabilities in Markov model for repetition and direct-follow pattern

[11]https://github.com/JochemVeldmanUU/Thesis_anomaly_generation/blob/main/Labelling%20anomalies%20pattern%201.ipynb

[12]https://github.com/JochemVeldmanUU/Thesis_anomaly_generation/blob/main/Labelling%20anomalies%20pattern%202.ipynb

[13]https://github.com/JochemVeldmanUU/Thesis_anomaly_generation/blob/main/Labelling%20anomalies%20pattern%203.ipynb

6.3.3 Case study: Generation of anomalies

In step two of the experiment setup, the generation of the anomalies is executed. This is done by applying the patterns designed in the second interview phase to the data. In the Markov model, we manipulate the probability of the relations displayed in the repetition, direct-follow and advanced repetition pattern figures. We increase the probability with different numbers and, for each addition, recalculate the probability so that the total probabilities sum up to one. In Table 14 all the changes in probability are displayed for the repetition and direct-follow pattern. In Table 15 the changed probabilities for the advanced repetition pattern are shown.

	Advanced repetition pattern			
	aanvragen ww - overzicht	aanvragen ww - home	overzicht - aanvragen ww	home - aanvragen ww
Deviation rate = 0	0.010	0.025	0.837	0.001
Deviation rate = 0.2	0.175	0.185	0.864	0.167
Deviation rate = 1	0.505	0.506	0.919	0.501
Deviation rate = 3	0.752	0.752	0.960	0.750
Deviation rate = 5	0.835	0.834	0.973	0.833
Deviation rate = 10	0.910	0.910	0.985	0.910
Deviation rate = 25	0.962	0.962	0.994	0.961

Table 15: Manipulation of probabilities in Markov model for advanced repetition pattern

We loop through the dictionary that contains the Markov model and store each session that includes activity /portal/page/portal/ home/diensten/aanvragen-ww in a list. For each session in the list, we generate a unique SessionID. The SessionID and the related activities are stored in a DataFrame, and this DataFrame is exported to a CSV file.

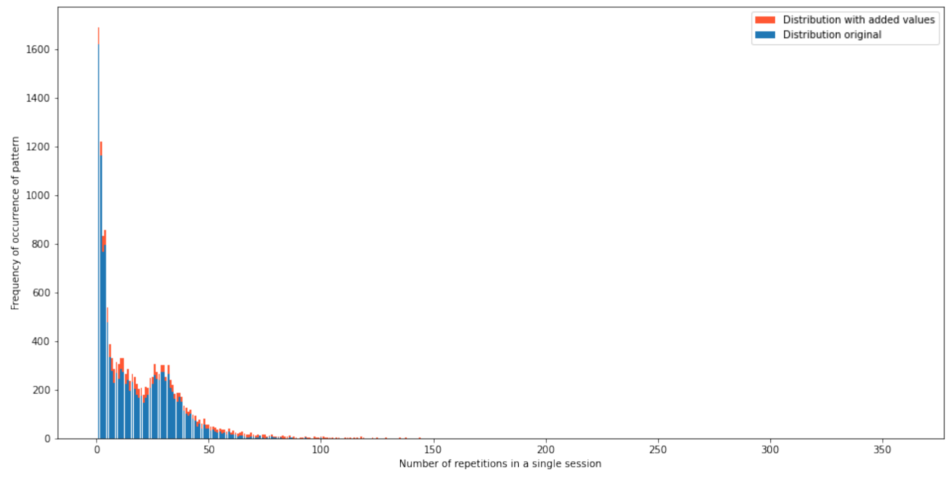


Figure 15: Distribution for repetition pattern with 2000 injected anomalous sessions and deviation rate = 5

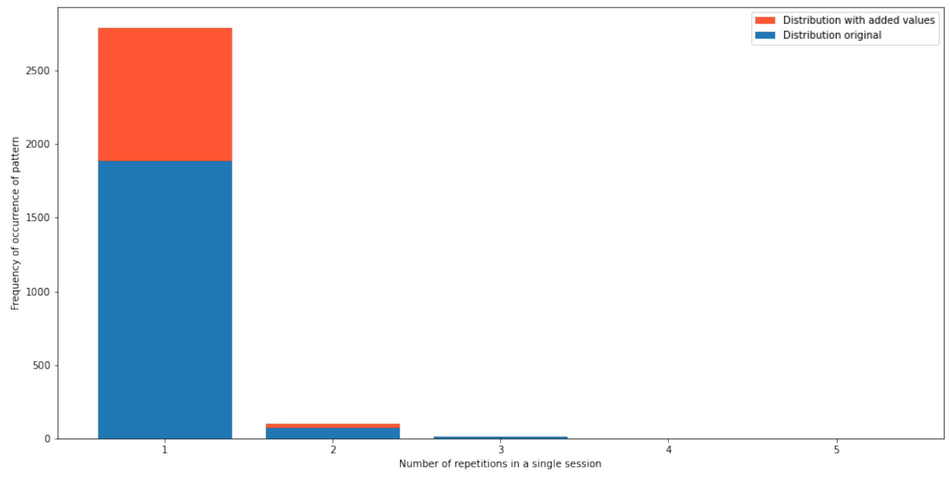


Figure 16: Distribution for direct-follow pattern with 2000 injected anomalous sessions and deviation rate = 5

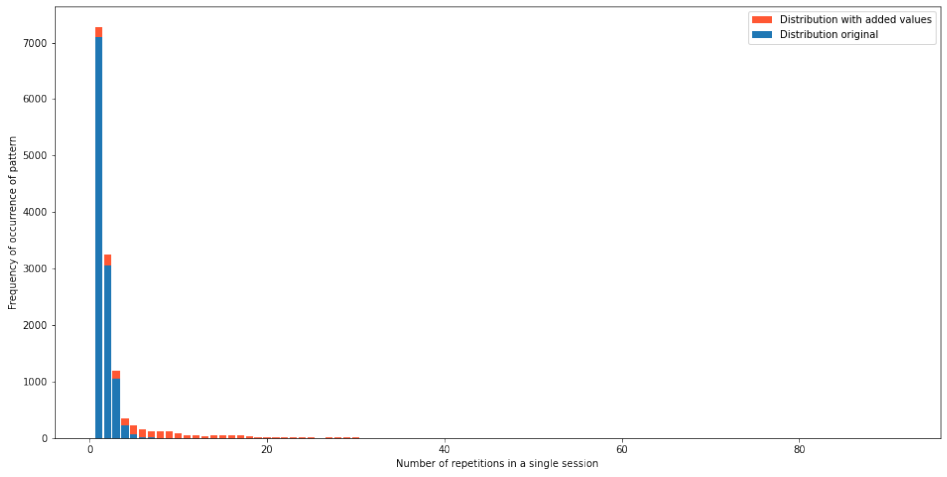


Figure 17: Distribution for advanced repetition pattern with 2000 injected anomalous sessions and deviation rate = 5

6.3.4 Case study: Sequence encoding for training and testing

Step three in the experiment setup includes preparing the data for training and testing the models. First, a selection of the data is made. This is done because the computer used for this experiment cannot handle the enormous amount of data that results from the encoding—the first 50.000 sessions in the dataset were selected.

Second, the data needs to be encoded since all the activities are in string format, and this cannot be used by the model we selected for this experiment. The encoding is done using one-hot encoding. One-hot encoding adds a new variable for each activity in the dataset, a dummy variable. The value of this variable represents the number of activities executed in a specific session, also known as bag-of-words encoding. Each session is encoded separately, and all available activities in the dataset are used. For the session labels, normal or anomaly, we have used a label encoder instead of the one-hot encoding. Labelencoder is used because it stores the encoded values in one variable, and unlike one-hot encoding, it does not create dummy variables. We chose to encode the target variable because it is easier to check how each category is encoded, as the trained model does not readily return how it encoded the target variable when the model automatically encodes it.

The normal data is split into a train and test set with 80% train and 20% test data. Ten anomaly sessions are included in the train data because this simulates an actual situation in the best possible way. In a real situation, there are no to very little known labelled anomalies. Assuming no anomalies in the data is impossible since this is never the case in an actual situation. The rest of the real anomalies are added to the test data. The generated anomalies are added to the training dataset in different quantities. The training and testing of the models are semantically shown in Figure 14.

6.3.5 Case study: Executing the models

Step four consists of running the models [14]. The model is trained on the train data and tested on the test data. The model we use for training and testing is a Logistic regression model. We use the default model from the package and only increase the max iteration parameter to 1000. This is done because the model does not converge to the optimum score if the default value for the max iterations is used. When the training of the model is done, we let the model predict the test data. The labels the model predicts on the test data are saved and used to evaluate the performance.

6.3.6 Case study: Evaluate model performance

In the last step of the case study, we evaluate the model's performance. We do this with the precision, recall and area under the curve metrics. The formula for the precision and recall are the following:

$$Precision\ score = \frac{True\ positives}{True\ positives + False\ positives}$$

$$Recall\ score\ (true\ positive\ rate) = \frac{True\ positives}{True\ positives + False\ negatives}$$

Precision measures how many cases that were predicted as positive are positive. Precision is essential in cases where it is expensive to be wrong. This can be the case, for example, in fraud detection. In fraud detection, it is time-consuming and expensive to investigate a potential fraud case. From the point of view of company profit, it may be wiser to let the potential fraud case go rather than waste resources investigating it.

Recall measures how many instances of the positives class have been captured correctly by the model. A high recall is preferable in cases where the cost of leaving a positive class undetected is high, and it is better to be safe than sorry. For example, in the case of a cancer diagnosis, it is better to say that someone has cancer and then discover in subsequent tests that someone does not have cancer than to say that someone does not have cancer when they do and let cancer grow worse without treatment.

[14]https://github.com/JochemVeldmanUU/Thesis_anomaly_generation/blob/main/Running_the_models.ipynb

$$F1\ score = 2\frac{Precision * \ Recall}{Precision\ +\ Recall}$$

The Precision and recall metrics can be combined in the F1 score. This F1 score is the harmonic mean of precision and recall. Since it takes both the metrics into account, it is a more robust metric than only looking at the precision or recall individually.

We also use the area under the curve (AUC) metric. The AUC score is a commonly used metric for detecting anomalies. Finally, we make use of the area under the curve of the receiver operating characteristic (ROC) curve. The ROC is a graph where the true positive rate is plotted against the false-positive rate for various threshold settings.

$$False\ positive\ rate = \frac{False\ positives}{False\ positives + True\ negatives}$$

For all three criteria, the higher the score, the better. For calculating the metrics we make use of the built in functions of Sci-kit learn. For the precision and recall this is the *.classification_report()* and for the AUC score this is *metrics.roc_curve()* and *metrics.auc()*.

6.4 Chapter summary

In this chapter, we started with a description of the data provided by the UWV. We have shown multiple distributions that show which activities are potential candidates for the execution of the use case and which one are not. Next, the results of the interviews where discussed. With the results from the data description and the interviews we have applied the three patterns to the domain. Finally, we have explained all the steps in the use case and how we have applied the designed patterns in the domain. In the following chapter we will discuss the results obtained in the experiment.

7 Results

In this section, we present the results obtained by the models with different amounts of injected anomalous sessions and different degrees of deviation rate of the generated anomalies. We first present the AUC scores of the models and then the precision and recall scores.

7.1 AUC scores

7.1.1 Repetition pattern

The achieved AUC scores for the repetition pattern can be seen in Table 16. In the table, we have put the different levels of deviation ratio in the columns and the number of the injected anomalous session as the row indexes. The first row consists of the same AUC values and can be seen as the baseline AUC score. Next, the table shows the AUC scores for different numbers of injected anomalies. The numbers in this table are a selection of all the configurations we ran. The AUC scores for all the configurations can be seen in Appendix D.

Number of anomalous sessions injected	AUC score deviation rate = 0	AUC score deviation rate = 0.2	AUC score deviation rate = 1	AUC score deviation rate = 5	AUC score deviation rate = 25
0	0.666	0.666	0.666	0.666	0.666
10	0.662	0.649	0.680	0.698	0.670
100	0.796	0.831	0.859	0.832	0.760
500	0.998	0.987	0.984	0.954	0.880
1000	0.995	0.997	0.993	0.978	0.917
2000	0.994	0.994	0.995	0.992	0.953

Table 16: AUC scores of models trained with different quantities of anomalies and different types of anomalies for repetition pattern.

The values from the table are visualised in Figure 18. On the x-axis, the number of injected anomalies is plotted, and on the y-axis, the AUC scores are presented. We have a separate line for all the different deviation ratios. The figure shows that all different manipulations follow the same trend and converge to the same AUC score. The deviation rate with a value of 25 is an exception to this trend, and its performance is lower than the other configurations.

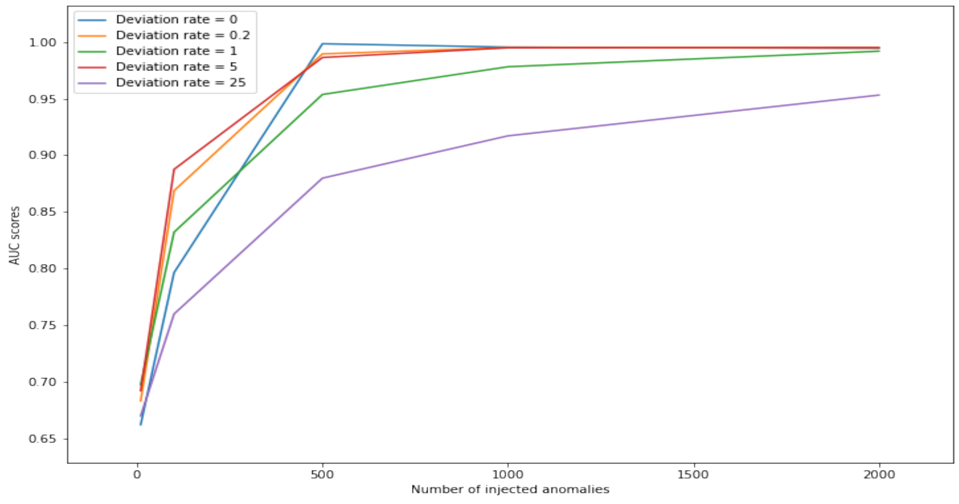


Figure 18: Selection AUC scores from repetition pattern plotted in a line chart

The table contains the AUC scores in more detail and shows that the AUC for the repetition pattern goes from 0.666 to 0.995, an increase of 0.329. This increase is obtained with a deviation rate of 1, which obtains the highest AUC score. On average, the different setups in the table achieve an increase of AUC of 0.320.

As seen in the figure, the AUC score of the deviation rate with a value of 25 does not converge to the same AUC score as the other manipulations. Instead, this deviation rate achieves an AUC score of 0.953, an increase of 0.287 compared to the baseline. This increase is worse than the average increase obtained by the different setups.

7.1.2 Direct-follow pattern

The AUC scores obtained for the direct-follow pattern can be seen in Table 17 and Figure 19. The setup of the table and figure are the same as the figure and table for the repetition pattern. The table contains the detailed AUC scores, and the figure contains a visualisation of the AUC scores.

Number of anomalous sessions injected	AUC score deviation rate = 0	AUC score deviation rate = 0.2	AUC score deviation rate = 1	AUC score deviation rate = 5	AUC score deviation rate = 25
0	0.623	0.623	0.623	0.623	0.623
10	0.636	0.636	0.649	0.649	0.688
100	0.630	0.643	0.753	0.773	0.708
500	0.817	0.869	0.914	0.940	0.940
1000	0.907	0.946	0.972	0.998	0.985
2000	0.944	0.976	0.995	0.996	0.996

Table 17: AUC scores of models trained with different quantities of injected anomalous session and different types of anomalies for direct-follow pattern.

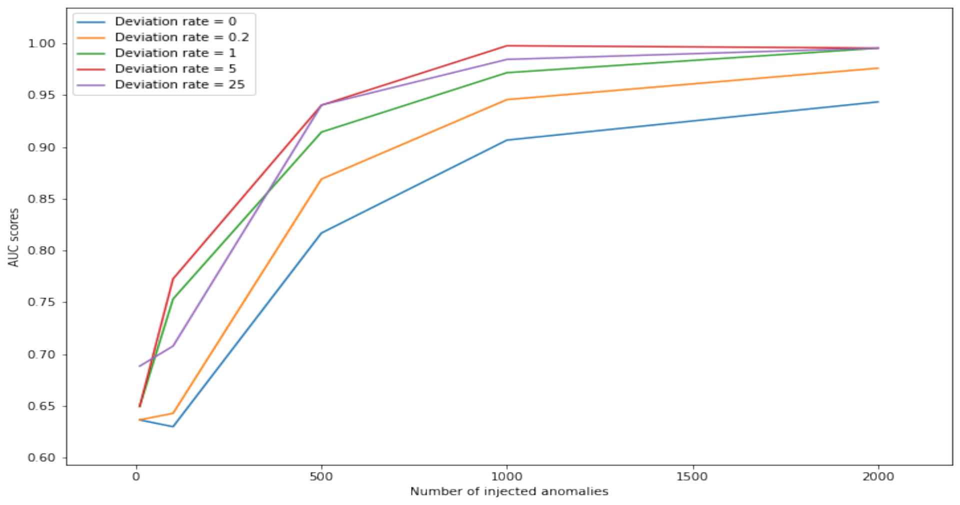


Figure 19: Selection AUC scores from direct-follow pattern plotted in a line chart

The table shows that the AUC for the direct follow pattern goes from 0.623 to 0.996, an increase of 0.373. This increase is obtained in the configurations with a deviation rate of 5 and 25. These two configurations also achieve the highest AUC score. On average, the different setups in the table achieve an increase of AUC of 0.358.

As seen in the figure, the AUC scores with a deviation rate of 0 and 0.2 do not converge to the same AUC score as the other deviation ratios. The deviation rate with a value of 0 achieves an AUC score of 0.944, an increase of 0.321 compared to the baseline. The deviation rate with a value of 0.2 achieves an AUC score of 0.976, an increase of 0.353. Both configurations achieve an AUC increase lower than the average

7.1.3 Advanced repetition pattern

The AUC scores obtained with the advanced repetition pattern can be seen in Table 18 and Figure 20. The setup of the table and figure are the same as the figure and table for the repetition and direct-follow pattern. The table contains the detailed AUC scores, and the figure contains a visualisation of the AUC scores.

Number of anomalous sessions injected	AUC score deviation rate = 0	AUC score deviation rate = 0.2	AUC score deviation rate = 1	AUC score deviation rate = 5	AUC score deviation rate = 25
0	0.577	0.577	0.577	0.577	0.577
10	0.569	0.572	0.560	0.566	0.556
100	0.621	0.607	0.588	0.577	0.560
500	0.823	0.640	0.612	0.596	0.570
1000	0.936	0.670	0.620	0.597	0.580
2000	0.980	0.710	0.636	0.613	0.572

Table 18: AUC scores of models trained with different quantities of injected anomalous session and different types of anomalies for advanced repetition pattern

The table shows that the AUC for the advanced repetition pattern goes from 0.577 to 0.980, an increase of 0.403. This increase is obtained with a deviation rate of 0, which obtains the highest AUC score. On average, the different setups in the table achieve an increase of AUC of 0.125.

As seen in the figure, the performance difference between the deviation rate with a value of and the other deviation ratios. The obtained AUC scores decrease for every increase in the deviation rate of the injected anomalies. For the deviation rate with a value of 25, four of the obtained AUC scores are lower than the baseline AUC. Other setups also achieve scores lower than the baseline, but not as much as the deviation rate with a value of 25 does.

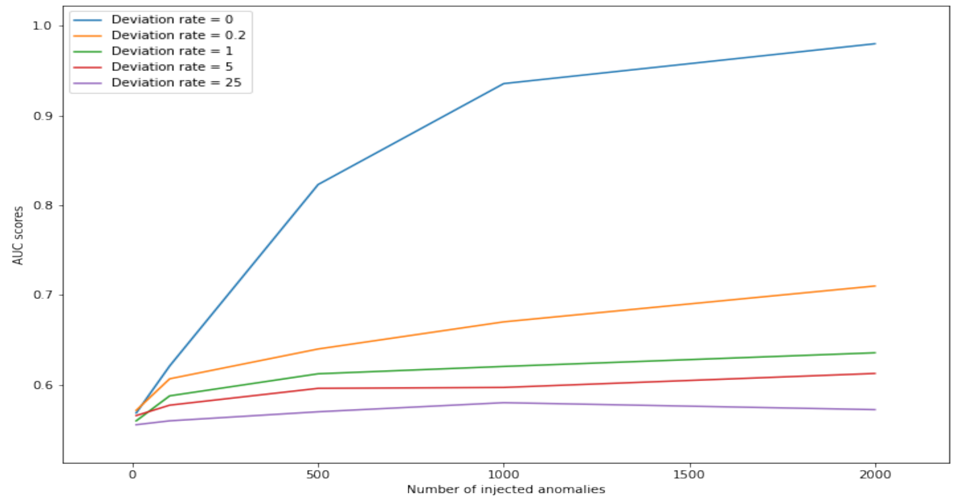


Figure 20: Selection AUC scores from advanced repetition pattern plotted in a line chart

7.2 Precision and recall scores

This subsection discusses the effect of the deviation rate and the number of injected anomalous sessions on the precision and recall scores. The precision and recall scores are shown in a table for every pattern. These tables show the precision and recall for the different numbers of injected anomalous sessions and deviation rates. The tables also include the precision and recall scores if no generated anomalies are injected. These scores serve as the baseline for comparing the different models and the performance they achieve. Next, we picked one deviation rate value and visualised all the precision and recall scores obtained in this specific manipulation for the different numbers of injected anomalies. The remaining figures can be seen in Appendix E. In these figures, we plotted the precision and recall scores on the y-axis and the number of injected anomalous sessions on the x-axis. These plots thus show the effect of increasing the injected anomalies on the precision and recall scores. The third kind of plot we have made is a precision and recall curve. These plots are made to compare the performance of the different configurations. In these plots, the recall score is plotted on the x-axis, and the precision score is plotted on the y-axis.

7.2.1 Repetition pattern

The precision and recall scores for the repetition pattern can be seen in Table 19. We can see a similar trend for all the values in the table. When we increase the number of injected anomalous sessions, the precision decreases and the recall increases. Increasing the deviation rate of the generated anomalies for the repetition pattern results in a slighter decrease in precision and a less increase in recall. On average, the precision values displayed in the table decrease by 0.11 and the recall increases by 0.65.

	Deviation rate = 0		Deviation rate = 0.2		Deviation rate = 1		Deviation rate = 5		Deviation rate = 25	
Number of anomalous sessions injected	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
0	0.99	0.33	0.99	0.33	0.99	0.33	0.99	0.33	0.99	0.33
10	1.00	0.32	0.99	0.37	1.00	0.38	0.99	0.40	0.99	0.34
100	0.99	0.59	0.99	0.74	0.99	0.78	0.99	0.66	1.00	0.52
500	0.93	1.00	0.93	0.98	0.94	0.98	0.96	0.91	0.99	0.76
1000	0.87	1.00	0.93	0.99	0.88	1.00	0.92	0.96	0.97	0.84
2000	0.80	1.00	0.93	0.99	0.82	1.00	0.90	0.99	0.94	0.91

Table 19: Precision and recall scores for different configurations for the repetition pattern

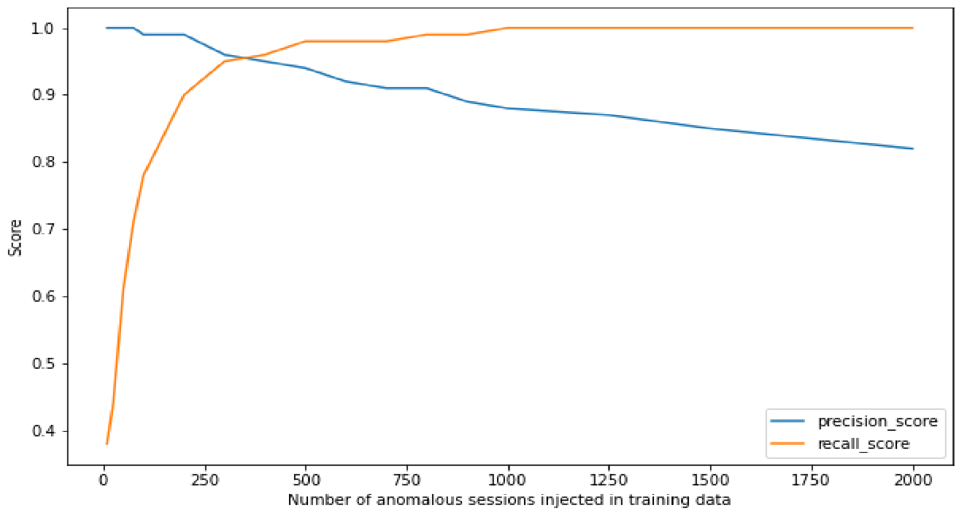


Figure 21: Precision and recall scores for repetition pattern with deviation rate value of 1

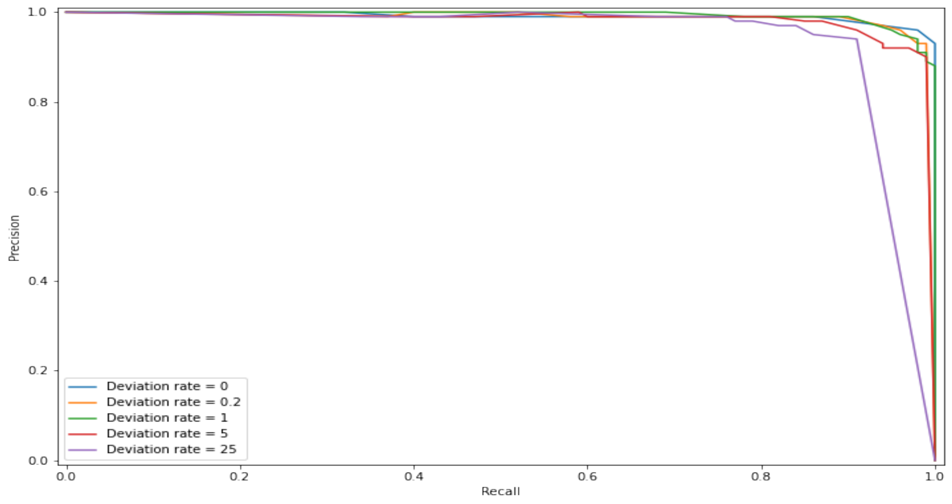


Figure 22: Precision and recall curves for different configurations of the repetition pattern

We have also visualised the trend of the precision and recall for the deviation rate with a value of 1 setup in Figure 21. This setup was chosen since it achieves the highest increase in the AUC score for the repetition pattern. When injecting more anomalies into the training data, the precision goes from 0.99 to 0.82, which decreases performance by 0.18. Moreover, the recall goes from 0.38 to 1.0, increasing to 0.62. The precision and recall cross each other between the 300 and 400 injected anomalies with a score of 0.95. Finally, the best precision and recall scores obtained with a deviation rate of 1 are summarized in an F1 score of 0.96. For the baseline, the model achieves an F1 score of 0.50. We thus obtain an increase of 0.46 in the F1 score.

In Figure 22 we have plotted the precision and recall curve for five different setups. The different lines overlap for most of the scores. However, the line with a deviation rate of 25 goes less far into the upper right corner. It achieves a worse precision and recall score than the four other setups. The other four lines deviate so little from each other that we can say that their performance on the precision and recall metrics are equal. The plot shows that increasing the deviation rate of injected anomalies does not have a significant impact on the performance of precision and recall.

7.2.2 Direct-follow pattern

For the direct-follow pattern we have displayed the precision and recall scores in Table 20. We can see a similar trend for all the values in the table. When injecting more anomalous sessions, there is an improvement in recall and a decline in precision performance. Increasing the deviation rate of the injected anomalies does not affect the precision scores. For all the precision scores, it decreases from 0.90 to around 0.45. Increasing the deviation rate does, however, affect the model's performance on the recall score. Increasing the deviation rate improves the recall score from 0.90, obtained in the deviation rate = 0 and deviation rate = 0.2 manipulations, to 1.00 in the higher abnormalities. The precision decreases with 0.44, and the recall increases with 0.71.

	Deviation rate = 0		Deviation rate = 0.2		Deviation rate = 1		Deviation rate = 5		Deviation rate = 25	
Number of anomalous sessions injected	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
0	0.90	0.25	0.90	0.25	0.90	0.25	0.90	0.25	0.90	0.25
10	0.88	0.27	0.91	0.27	0.92	0.30	0.92	0.30	0.94	0.38
100	0.83	0.26	0.81	0.29	0.93	0.51	0.91	0.55	0.89	0.42
500	0.64	0.64	0.70	0.74	0.72	0.83	0.74	0.88	0.75	0.88
1000	0.56	0.82	0.59	0.90	0.61	0.95	0.63	1.00	0.60	0.97
2000	0.43	0.90	0.45	0.90	0.46	1.00	0.46	1.00	0.46	1.00

Table 20: Precision and recall scores for different configurations for the direct-follow pattern

In Figure 23 we have visualised the precision and recall for the deviation rate with a value of 5 since this setup achieved the highest AUC score. When injecting more anomalies, the precision goes from 0.90 to 0.46, increasing to 0.44. The recall goes from 0.25 to 1.00, an increase of 0.75. The precision and recall cross each other between the 300 and 400 injected anomalies with a score of 0.82. From the table, we derive that the best F1 score obtained by the deviation rate with a value of 5 is 0.77. For the baseline, the model achieves an F1 score of 0.39. We thus obtain an increase of 0.38 in the F1 score.

We plotted the precision and recall curve for five different setups to compare the different configurations. This plots can be seen in 24. This graph shows that the lines with deviation rates of 5 and 25 perform the best. These lines are nearest to the upper right corner, indicating good performance. The deviation rates of 5 and 25 do not show much difference; thus, both setups achieve the same performance. The other three lines underperform, with the deviation rate of 0 having the worst performance. The plot shows that increasing the deviation rate of injected anomalies results in better precision and recall

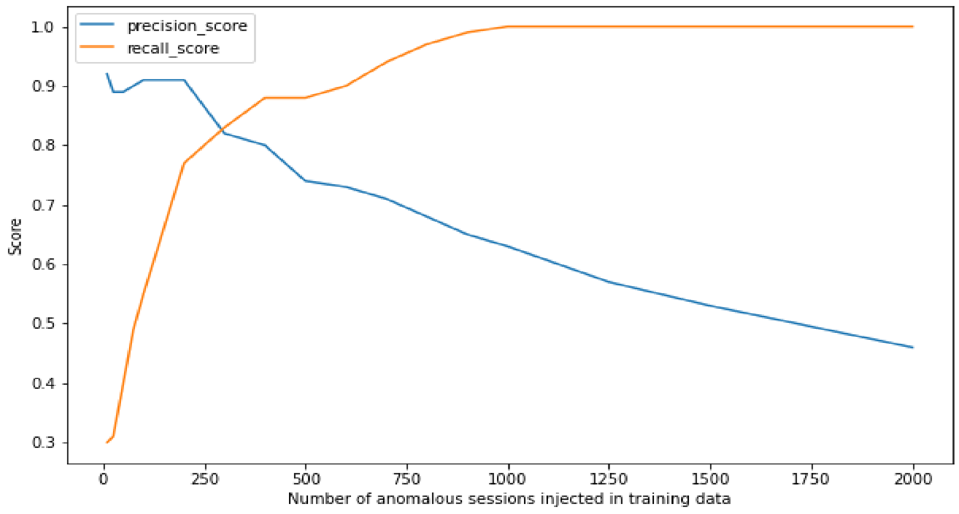


Figure 23: Precision and recall scores for direct-follow pattern with deviation rate value of 5

scores.

7.2.3 Advanced repetition pattern

Precision and recall scores obtained with the advanced repetition pattern can be seen in Table 21. The values show that increasing the number of injected anomalous sessions results in a lower precision and a higher recall. Rising the deviation rate of the anomalies results in a worse performance of the different setups. For the recall metric, the precision values stay the same for the different setups. For the setup with a deviation rate of 0, the precision decreases from 0.98 to 0.76, decreasing by 0.22. Moreover, the recall increased from 0.15 to 0.97, an increase of 0.72. The setup with a deviation rate of 25 shows different results. The decrease in precision is the same, but the recall does not improve. We can see that the precision decreases with 0.22, and recall increases with 0.26.

We have visualised the precision and recall for the probability with a deviation rate of 0 in Figure 25. The setup with a deviation rate of 0 achieved the best AUC score. The precision and recall cross when 900 to 1000 anomaly sessions are injected. The precision and recall are 0.84 at this moment. From the table, we derive that the best F1 score obtained by the advanced repetition pattern is 0.85. For the baseline, the model achieves an F1 score of 0.26. We thus obtain an increase of 0.59 in the F1 score.

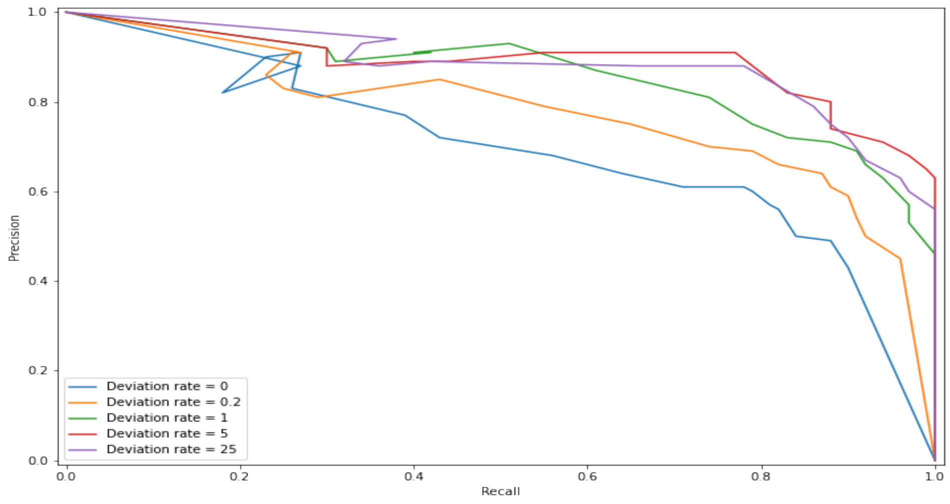


Figure 24: Precision and recall curves for different configurations of the direct-follow pattern

In Figure 26 we have plotted the precision and recall curve for five different setups. The plot shows that the deviation rate with a value 0 probability achieves the best scores since it is the line closest to the upper right corner. The other setups perform worse and do not reach high in the upper right corner. The deviation rate with a value of 0.2 setup is the second-best performing configuration. The other three setups are equally bad. The plot shows that increasing the deviation rate of injected anomalies results in worse performance measured in precision and recall.

	Deviation rate = 0		Deviation rate = 0.2		Deviation rate = 1		Deviation rate = 5		Deviation rate = 25	
Number of anomalous sessions injected	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
0	0.98	0.15	0.98	0.15	0.98	0.15	0.98	0.15	0.98	0.15
10	0.96	0.14	0.96	0.14	0.95	0.12	0.96	0.13	0.97	0.11
100	0.92	0.24	0.99	0.21	0.98	0.18	0.93	0.15	0.95	0.12
500	0.89	0.65	0.91	0.28	0.93	0.23	0.88	0.19	0.91	0.14
1000	0.84	0.88	0.87	0.34	0.82	0.24	0.79	0.20	0.89	0.16
2000	0.76	0.97	0.79	0.42	0.73	0.27	0.74	0.23	0.76	0.15

Table 21: Precision and recall scores for different configurations for the advanced repetition pattern

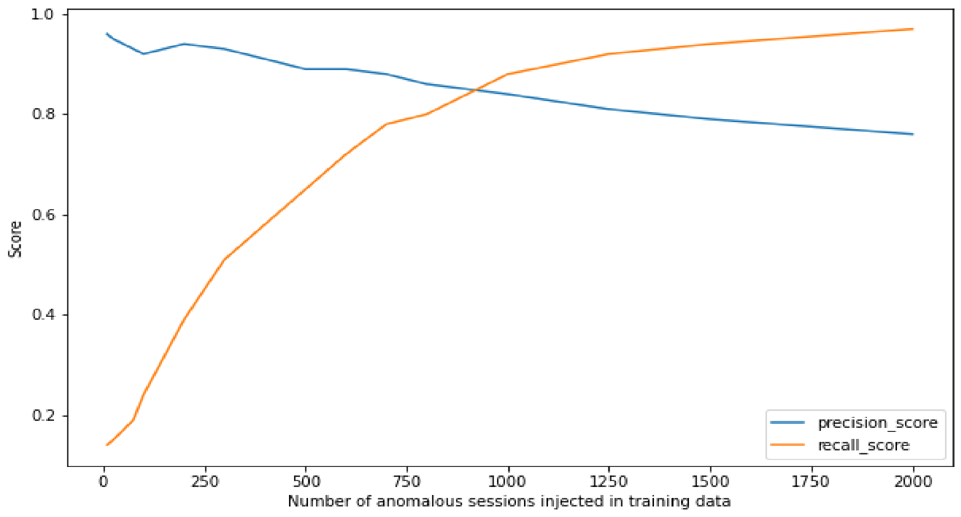


Figure 25: Precision and recall scores for advanced repetition pattern with deviation rate with a value of 0

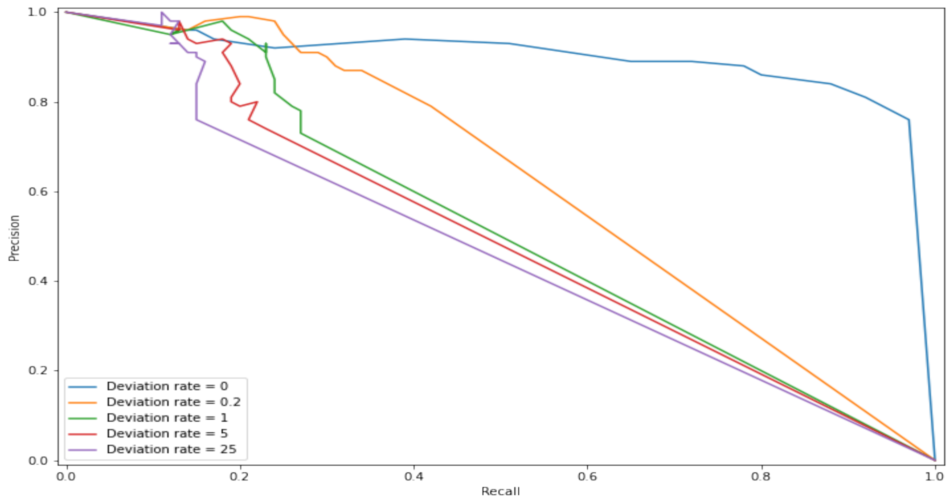


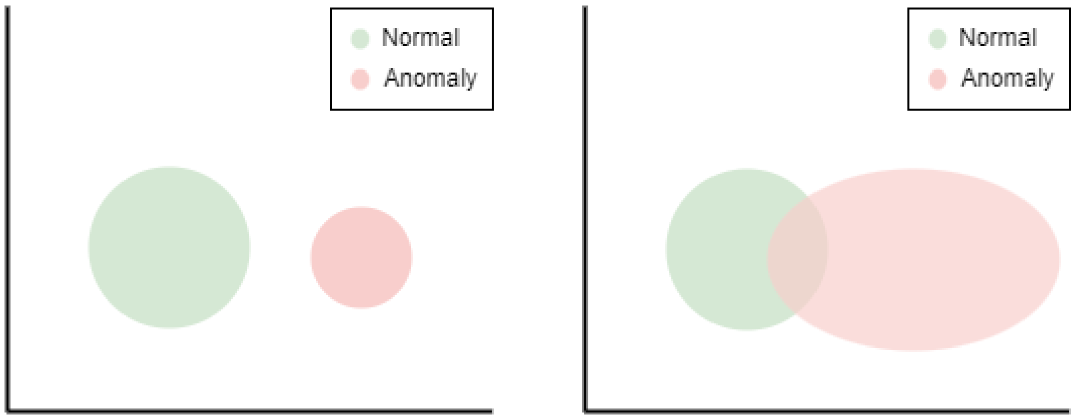
Figure 26: Precision and recall curves for different configurations of the advanced repetition pattern

8 Discussion

In the discussion section, we discuss the results obtained in this research and debate the contribution and implications of the research and the proposed method.

8.1 Discussing of results

8.1.1 Quantitative results



(a) Situation with only a few anomalies injected (b) Situation with a lot of anomalies injected

Figure 27: Comparison of clusters with a few injected anomalies and with a lot of injected anomalies

If we look at the generation of anomalies based on the three patterns, we see that for the repetition and direct-follow patterns, generating and injecting the anomalies helps to improve the detection accuracy significantly. For the repetition pattern, the generating and injecting achieves an increase of an AUC from 0.666 to 0.995 and an increase in the F1 score from 0.50 to 0.96. For the direct-follow pattern, the AUC increases from 0.623 to 0.996 and the F1 score from 0.39 to 0.77. For the advanced repetition pattern, the injection of anomalies did not help as much as for the other two patterns. It only achieves good results when the deviation rate is 0. For this deviation rate, we achieve an increase in AUC from 0.577 to 0.980 and in F1 score from 0.26 to 0.85.

We found that the AUC score and recall increase when injecting more anomalies, and precision decreases for all the different patterns. We argue that the answer to this phenomenon can be found in the definition of an anomaly—the concept of an anomaly in that it is a data instance that is abnormal from normal behaviour. When injecting more generated anomalies, the number of anomalies in the train data increases and become so many and diverse that they start to overlap with the normal behaviour. In addition, we must consider that anomaly generation is a semi-random process. This semi-random process makes the generated anomalies all differ from each other and thus diverse. Adding a lot of the generated anomalies is thus adding a large variety of anomalies to the data. This creates a big cluster of generated anomalies that overlaps with the cluster of normal behaviour. Adding more generated anomalies thus makes it harder to distinguish the real normal behaviour from the injected anomalies and results in a detection models that is not able to distinguish all the normal and anomaly instances. These two reasons are why the precision decreases and the recall increases when more generated anomalies are injected. Figure 27 shows a visualisation of the clusters.

Increasing the deviation rate does not significantly affect the AUC scores obtained by the models for the repetition and direct-follow patterns. Both patterns achieve an AUC score that converges to roughly the same value for all different configurations with different deviation rates. We can see a decrease in AUC for the advanced repetition pattern when we increase the deviation rate parameter.

The effect of increasing the deviation rate on the precision and recall scores is neutral for the repetition pattern. All the configurations perform the same. For the direct-follow, we can see that increasing the deviation rate results in better performance. On the other hand, we can see a decrease in the precision and recall scores for the advanced repetition.

When looking at the lacking performance of the advanced repetition pattern when we increase the deviation rate, we argue that this is due to the complexity of the pattern and our lack of domain knowledge. The complexity of the pattern requires a deeper understanding of how the activities are related to each other for a successful generation of new anomalies, and we did not have this domain expertise. If we analyse the generated sessions for this pattern, we can see that the sessions are relatively short and contain a low number of activities. This is different from the actual anomalies. Looking at the distribution with the added anomalies for the advanced repetition pattern in Figure 17, we can see that the generated anomalies are not in line with the existing distribution. We assume that these two arguments are the reason for the poor scores for the advanced repetition pattern.

For the other two patterns, we have concluded that they performed well. Analysing the generated sessions shows that they overlap well with the original anomalies. This can be seen in Figure 15 and Figure 16. We can only manipulate one probability for these two patterns, making the generation easier to control and thus easier to generate anomalies that are as closest to the domain as possible.

8.1.2 Qualitative results

We have obtained a wide variety of anomaly characteristics by doing extensive literature research and have combined all these anomaly characteristics in a taxonomy. The taxonomy consists of five domains and more than forty characteristics. More could probably have been found and added when searching for an extended period. This could be especially the case if we had considered more domains in the research. However, we have stayed with the domains and the found characteristics since we examined the taxonomy as sufficient. This choice is based on a trade-off between time and the found characteristics. In theory, more characteristics could have been found, but this would have been at the expense of the rest of the research because a time limit bound us.

Based on this taxonomy, we have designed the three general patterns. Since they are built from a taxonomy based on extensive literature research, we have high confidence in the designed patterns.

8.2 Additional results

For additional results, we ran the same experiment but with an alternative method commonly used when there is a lack of labelled data. This is done to compare our method with an already existing method. We have also used two other models to see how well the generation of anomalies with Markov models can be generalized to other models.

8.2.1 Additional results oversampling implementation

For the additional results, we have used .RandomOverSampler() from the Imbalanced learn library [15]. This function oversamples the minority class by picking samples at random with replacement. The original trainset includes ten anomaly sessions and 39.957 normal sessions. The oversampling function increases the anomaly sessions to 39.957, equal to the number of normal sessions. The resulting train set from the oversampling is used to train a logistic regression model, and this model is used for prediction on the test set.

8.2.2 Additional results oversampling comparison

In the results in Table 22, we can see the results obtained by the logistic regression with the oversampling data. The bold scores are the highest when we compare oversampling with the Markov model. Compared to the generation of anomalies with the Markov model,

[15] `https://github.com/JochemVeldmanUU/Thesis_anomaly_generation/blob/main/Oversampling.ipynb`

	AUC		Precision		Recall		F1	
Pattern	Oversampling	Markov	Oversampling	Markov	Oversampling	Markov	Oversampling	Markov
Repetition	0.888	**0.995**	**0.99**	0.93	0.78	**0.99**	0.87	**0.96**
Direct-follow	0.870	**0.996**	**0.89**	0.72	0.74	**0.83**	**0.81**	0.77
Advanced repetition	0.757	**0.980**	**0.92**	0.76	0.51	**0.97**	0.66	**0.85**

Table 22: AUC, precision and recall scores with oversampling technique

we can see that oversampling achieves a better precision score but a lower recall than the Markov model. Looking at both scores combined in the F1 score, we can see that the Markov model achieves a higher score for the repetition and advanced repetition scores. In addition, the oversampling technique achieves a better F1 score for the direct-follow pattern. Depending on the applied deviation rate, Markov models can reach a higher precision or recall than the scores obtained with the oversampling technique, but this goes at the cost of the other metric.

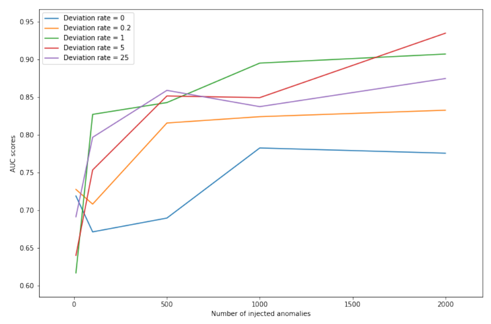
Looking at the AUC score, we can see that the oversampling technique achieves lower AUC scores than the models trained with the generated anomalies from the Markov model do. For example, with the Markov model, the highest AUC for repetition is 0.995, for direct-follow 0.996, and for advanced repetition is 0.980. This means that with oversampling, the repetition pattern achieves a 0.107 lower score, direct-follow achieves a 0.126 lower score, and advanced repetition achieves a 0.223 lower AUC score. Thus, we can conclude that the generation with the Markov model achieves better AUC scores for all three patterns than the generation with the oversampling technique.

8.2.3 Additional results Decision tree

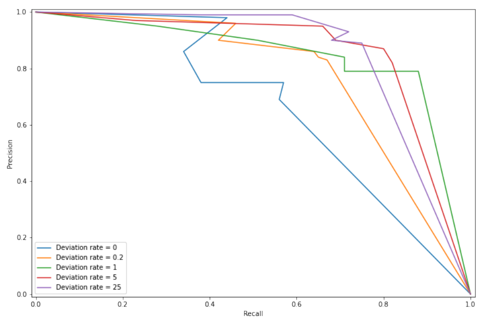
We ran the same experiment with a Decision tree instead of a Logistic regression model. These can be seen in Figure 28, 29 and 30. The results achieved with the Decision tree model are not as good as with the Logistic regression. However, the results show the same trend as the results from the Logistic regression model. In the AUC curves, we can see that increasing the number of injected anomalous sessions increases the performance. Increasing the deviation rate also increases the performance slightly. This increase in AUC when the deviation rate increase is different when compared with the Logistic regression. In addition, the Decision tree also achieves good results for the advanced repetition pattern when the deviation is increased, which is not the case for the Logistic regression model.

We can also see a similar trend for the precision and recall curves. When the number of injected anomalous sessions increases, the precision decreases and the recall increases. Different, however, is that the performance of the Decision tree on the precision and recall metrics for all the three different patterns increases when the deviation rate is increased.

For the Logistic regression, this is only the case for the direct-follow pattern.

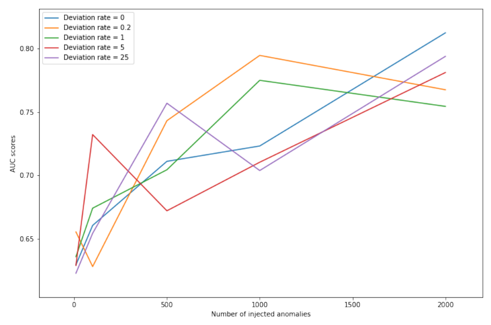


(a) AUC curves for repetition pattern and Decision tree

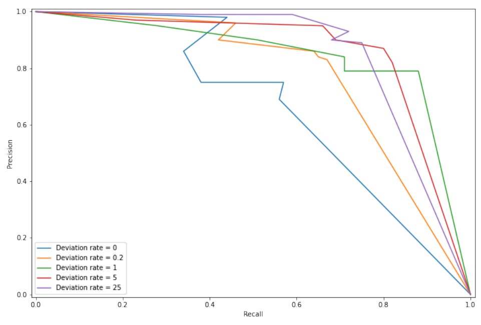


(b) Precision and recall curves for repetition pattern and Decision tree

Figure 28: AUC and precision and recall curves for repetition pattern and Decision tree model

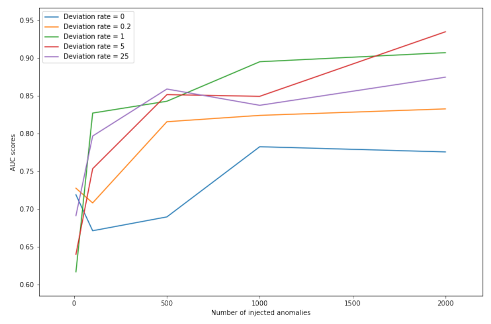


(a) AUC curves for direct-follow pattern and Decision tree

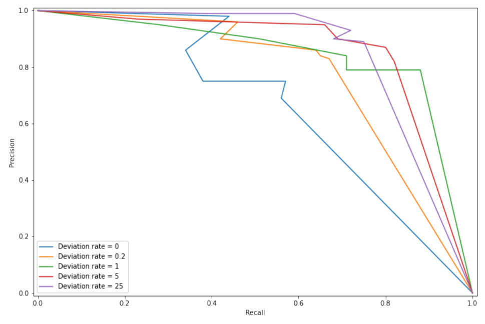


(b) Precision and recall curves for direct-follow pattern and Decision tree

Figure 29: AUC and precision and recall curves for direct-follow pattern and Decision tree model

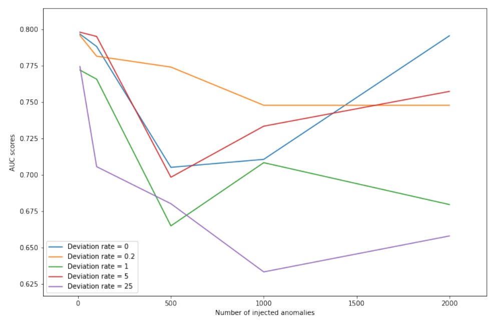


(a) AUC curves for advanced repetition pattern and Decision tree

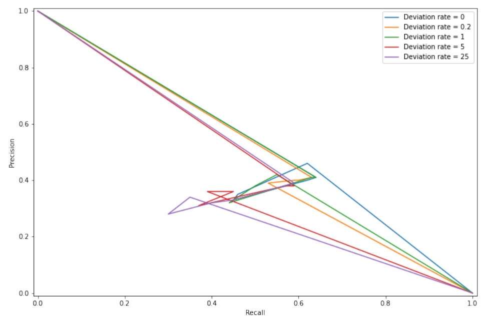


(b) Precision and recall curves for advanced repetition pattern and Decision tree

Figure 30: AUC and precision and recall curves for advanced repetition pattern and Decision tree model



(a) AUC curves for repetition pattern and Isolation forest



(b) Precision and recall curves for repetition pattern and Isolation forest

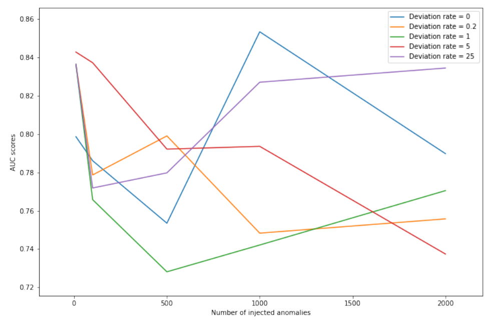
Figure 31: AUC and precision and recall curves for repetition pattern and Isolation forest

8.2.4 Additional results Isolation forest

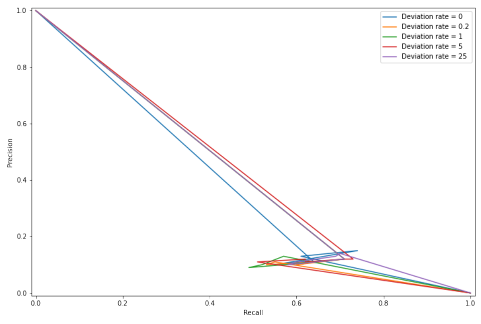
Isolation forest is an unsupervised model and is added to the additional results to compare the results of the generation method for supervised and unsupervised models. We can see the results of the Isolation forest model in Figure 31, 32 and 33.

The quality of the results obtained with the Isolation forest model differs from those obtained with the Logistic regression and Decision tree models. We cannot distinguish a real trend in the AUC curves. They increase and decrease when the number of injected anomalous sessions is increased. Due to this performance behaviour, it cannot be said what the best model is, and we can also not conclude the effect of the deviation rate parameter on the performance.

Looking at the precision and recall curves, we can see that for all three patterns there is an almost straight diagonal line from the left upper corner to the right lower corner. This shows that the Isolation forest model cannot distinguish the normal instances from the anomaly instances. Therefore, it can either achieve high precision and a low recall or a high recall and low precision.



(a) AUC curves for direct-follow pattern and Isolation forest

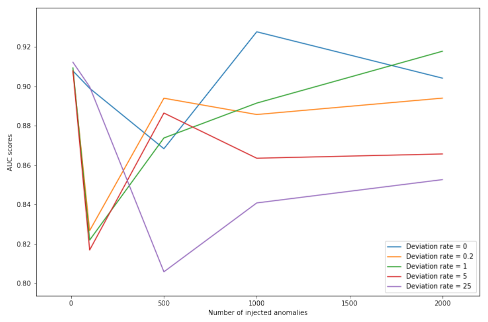


(b) Precision and recall curves for direct-follow pattern and Isolation forest

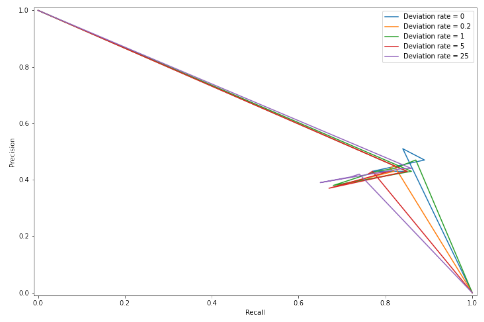
Figure 32: AUC and precision and recall curves for direct-follow pattern and Isolation forest

8.3 Implication

In the section on implication, we discuss the meaning of the results obtained. The taxonomy implies that other people, researchers and practitioners, can use the taxonomy to construct patterns that apply in their domain. The repetition and direct-follow pattern can be applied directly in other domains since they have been proved successful. The advanced repetition pattern can also be used, but without the deviation rate parameter, or the Decision tree model should be used instead of a Logistic regression. Generating anomalies based on these patterns allowed for an alternative labelling strategy for the data. The proposed approach can be used when there is a lack of labels. Using the code that is used in this research and that is made public using the approach can be an easy task. It does



(a) AUC curves for advanced repetition pattern and Isolation forest



(b) Precision and recall curves for advanced repetition pattern and Isolation forest

Figure 33: AUC and precision and recall curves for advanced repetition pattern and Isolation forest

take time to understand the domain and determine the relevant features, but this is nothing compared to manually labelling all data. Correctly applied, our method is a suitable alternative for obtaining labelled data ready to be adapted to other domains.

8.4 Contribution

This research approach is created to offer a possibility to obtain labelled data in an alternative way. Its most significant contribution is that it has shown that the alternative labelling strategy is effective and can be applied in practice. Besides this contribution, the research also contributes to establishing a taxonomy of anomaly characteristics, in which it distinguishes general process mining anomaly characteristics and domain-specific anomaly characteristics. The designed patterns that result from this taxonomy are the third contribution of this research. Building a Markov chain from the original dataset is a convenient method to incorporate the patterns and generate the anomalous sessions. Another contribution is that we have shown the effect of increasing anomalies' abnormality with the deviation rate and the number of injected anomalies. Our final contribution is that we used the UWV dataset from the BPIC 2016 and proved that this dataset could be used for research other than what it was intended for when it was published.

8.5 Future work

For future work, we suggest two options that can be researched. The first is applying the approach, repetition, and direct-follow pattern to another dataset. The second suggestion is to construct new patterns based on the taxonomy and to use these patterns on the UWV dataset. The final suggestion is to apply more models and research the performance of unsupervised models and the effect of injected anomalous instances more in detail.

9 Conclusion

The conclusion section answers the sub-research questions and the main research question. So, first, we answer the sub-research questions and second, the main research question.

9.1 Sub research questions

9.1.1 Sub-question 1

What fraud characteristics are described in the scientific literature?

We have conducted a structured literature review to establish the fraud characteristics and discussed the results in section 3. We have found 41 characteristics, which we can distinguish into 16 more general and 25 detailed characteristics. The general characteristics are the process mining characteristics. We can distinguish high-level process mining characteristics and low-level characteristics in the process mining characteristics. The more detailed characteristics we have found are from the credit card, e-commerce and bot detection domain. We have also found characteristics that we classified as remaining fraud characteristics. For each domain, we have listed multiple detailed anomaly characteristics. We have described how each characteristic translates to an event log for the detailed process mining characteristics. Based on these translations, we link the process mining characteristics to the detailed process mining characteristics.

9.1.2 Sub-question 2

How can fraud characteristics be used to generated anomalous sessions?

We have formulated an answer for this sub-question in the approach section. We have found that generating anomalies with a Markov chain and applying general anomaly patterns to the Markov chain is a suitable way to generate anomalies. The Markov chain is built from the whole event data and includes all the possible behaviour in the dataset. Therefore, the probabilities in the Markov chain can easily be manipulated, and we do this with the help of three designed patterns that we apply to the UWV domain. The patterns that we have designed are repetition, direct-follow and advanced repetition.

9.1.3 Sub-question 3

To what extent would the generation and injection of anomaly behaviour in an event log help to improve the detection performance?

Looking at the AUC and precision and recall scores, we can conclude that the generation of the repetition pattern has successfully improved the detection performance of the model. The AUC scores are good, and the precision and recall also converge to a high number.

The direct-follow pattern has also shown to be effective in improving detection performance. The AUC scores we obtain are high. The precision of the model also increases when we add more generated data. However, the precision score decreases when we add a higher number of anomalous sessions, but there is a clear optimum for this pattern where there is a high precision, recall and AUC.

The last pattern, advanced repetition, has shown to be successful when not changing the deviation parameter. The AUC scores obtained with a deviation rate of 0 are high. This also applies to the precision and recall scores. However, changing the deviation rate significantly decreases all the achieved results.

Comparing the three different models used in this study, we can conclude that the performance of the supervised models improved when different sessions were injected. However, for the unsupervised method, further research is needed to determine how and why they perform as they do.

In general, we can conclude that the effect of a higher percentage of anomalies in the data is an increase in AUC and recall but a decline in precision. Regarding the type of anomalies, we can conclude that increasing the anomalies of an injected session has a limited positive impact on the AUC score. The effect on precision is that making sessions more anomalous increases the precision score but decreases the recall score.

9.2 Main research questions

How can anomalous sequences be generated to help train anomaly detection models to improve their performance?

In this research, we have proposed an approach for generating and injecting anomalous sequences into an event data file. This approach can be used to successfully generate anomalous sequences to help train anomaly, detection models.

We first established a taxonomy of potential anomaly behaviour to execute this approach. Based on the taxonomy, we have created three general patterns; repetition, direct-follow and advanced repetition. We have applied these general patterns to the domain of the UWV. To inject anomalous sessions based on the created patterns, we have built a Markov model of the event data from the UWV in which we can change every probability between activities. We can apply the three designed patterns in the Markov model to generate new anomalous sequences.

Applying this approach to the UWV dataset shows excellent results and proves that the approach is valuable and relevant. The use case with the UWV dataset demonstrates that the repetition and direct-follow pattern obtain excellent results but that the advanced repetition pattern falls short in AUC scores when increasing the deviation rate. However, when the amount of injected anomalous sessions is optimised, all three patterns receive good AUC, precision and recall scores.

We have also tested the effect of injecting more anomalies into the train data and making the anomalies more anomalous. Results show that increasing the quantity of injected anomalies improves the AUC and recall score but decreases the precision score. Raising the deviation rate of the anomalies by increasing the Markov model does not have a significant effect. In some cases, this increase results in better AUC scores, but all AUC scores converge to roughly the same maximum when more anomalies are injected into the train data. This offsets the potential effect of increasing the deviation rate of an anomaly for the AUC score. However, increasing the deviation rate of anomalies does have a positive effect on the precision score.

In total, we have used three different models in this study. Two with supervision and one without supervision. The performance of the two supervised models increased with the models we used in this study. However, the performance of the unsupervised model did not show a clear trend and further research is needed to draw conclusions about the effect of injecting generated anomalies into unsupervised models. We have also compared the generation with Markov models and the oversampling technique. The Markov model performs better on the AUC score and the recall metric. On the other hand, the oversampling technique achieves higher precision scores.

Concluding, we can say that we have developed a successful approach that can help train anomaly detection models. Furthermore, it has proved its value by being applied successfully in a use case with actual data, and it is ready to be applied in other contexts and domains.

References

[1] W.M.P. van der Aalst. *Process Mining - Data Science in Action, Second Edition*. Springer, 2016. ISBN: 978-3-662-49850-7. DOI: `10.1007/978-3-662-49851-4`. URL: `https://doi.org/10.1007/978-3-662-49851-4`.

[2] W.M.P. van der Aalst. "Process mining and simulation: a match made in heaven!" In: *Proceedings of the 50th Computer Simulation Conference, SummerSim 2018, Bordeaux, France, July 09-12, 2018*. ACM, 2018, 4:1–4:12. URL: `http://dl.acm.org/citation.cfm?id=3275386`.

[3] W.M.P. van der Aalst. "Process Mining: Overview and Opportunities". In: *ACM Trans. Manag. Inf. Syst.* 3.2 (2012), 7:1–7:17. DOI: `10.1145/2229156.2229157`. URL: `https://doi.org/10.1145/2229156.2229157`.

[4] W.M.P. van der Aalst, A. Adriansyah, A.K.A de Medeiros, F. Arcieri, T. Baier, T. Blickle, R.P.J. Chandra Bose, P. van den Brand, R. Brandtjen, J.C.A.M. Buijs, A. Burattin, J. Carmona, M. Castellanos, J. Claes, J. Cook, N. Costantini, F. Curbera, E. Damiani, M. de Leoni, P. Delias, B.F. van Dongen, M. Dumas, S. Dustdar, D. Fahland, D.R. Ferreira, W. Gaaloul, F. van Geffen, S. Goel, C.W. Günther, A. Guzzo, P. Harmon, A.H.M. ter Hofstede, J. Hoogland, J. Espen Ingvaldsen, K. Kato, R. Kuhn, A. Kumar, M. La Rosa, F.M. Maggi, D. Malerba, R.S. Mans, A. Manuel, M. McCreesh, P. Mello, J. Mendling, M. Montali, H.R.M. Nezhad, M. zur Muehlen, J. Munoz-Gama, L. Pontieri, J. Ribeiro, A. Rozinat, H.S Pérez, R.S. Pérez, M. Sepúlveda, J. Sinur, P. Soffer, M. Song, A. Sperduti, G. Stilo, C. Stoel, K.D. Swenson, M. Talamo, W. Tan, C. Turner, J. Vanthienen, G. Varvaressos, E. Verbeek, M. Verdonk, R. Vigo, J. Wang, B. Weber, M. Weidlich, T. Weijters, L. Wen, M. Westergaard, and M.T. Wynn. In: *Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I*. Vol. 99. Lecture Notes in Business Information Processing. Springer, 2011, pp. 169–194. DOI: `10.1007/978-3-642-28108-2\_19`. URL: `https://doi.org/10.1007/978-3-642-28108-2\_19`.

[5] *ABN AMRO betaalt 480 miljoen euro vanwege ernstige tekortkomingen bij het bestrijden van witwassen*. Apr. 19, 2021. URL: `https://www.om.nl/actueel/nieuws/2021/04/19/abn-amro-betaalt-480-miljoen-euro-vanwege-ernstige-tekortkomingen-bij-het-bestrijden-van-witwassen` (visited on 07/10/2022).

[6] R. Accorsi, T. Stocker, and G. Müller. "On the exploitation of process mining for security audits: the process discovery case". In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18-*

22, 2013. ACM, 2013, pp. 1462–1468. DOI: 10.1145/2480362.2480634. URL: https://doi.org/10.1145/2480362.2480634.

[7] I. Achituve, S. Kraus, and J. Goldberger. "Interpretable Online Banking Fraud Detection Based On Hierarchical Attention Mechanism". In: *29th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2019, Pittsburgh, PA, USA, October 13-16, 2019*. IEEE, 2019, pp. 1–6. DOI: 10.1109/MLSP.2019.8918896. URL: https://doi.org/10.1109/MLSP.2019.8918896.

[8] A.O. Adewumi and A.A. Akinyelu. "A survey of machine-learning and nature-inspired based credit card fraud detection techniques". In: *Int. J. Syst. Assur. Eng. Manag.* 8.2s (2017), pp. 937–953. DOI: 10.1007/s13198-016-0551-y. URL: https://doi.org/10.1007/s13198-016-0551-y.

[9] M. Ahmed, A.N. Mahmood, and M.R. Islam. "A survey of anomaly detection techniques in financial domain". In: *Future Gener. Comput. Syst.* 55 (2016), pp. 278–288. DOI: 10.1016/j.future.2015.01.001. URL: https://doi.org/10.1016/j.future.2015.01.001.

[10] L. Akoglu, R. Chandy, and C. Faloutsos. "Opinion Fraud Detection in Online Reviews by Network Effects". In: *Proceedings of the Seventh International Conference on Weblogs and Social Media, ICWSM 2013, Cambridge, Massachusetts, USA, July 8-11, 2013*. The AAAI Press, 2013. URL: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM13/paper/view/5981.

[11] M. Alizadeh, X. Lu, D. Fahland, N. Zannone, and W.M.P. van der Aalst. "Linking data and process perspectives for conformance analysis". In: *Comput. Secur.* 73 (2018), pp. 172–193. DOI: 10.1016/j.cose.2017.10.010. URL: https://doi.org/10.1016/j.cose.2017.10.010.

[12] S. Alla and S.K. Adari. *Beginning anomaly detection using python-based deep learning*. Springer, 2019.

[13] S. Alter. "Theory of Workarounds". In: *CAIS* 34 (2014), p. 55. DOI: 10.17705/1cais.03455. URL: https://doi.org/10.17705/1cais.03455.

[14] P. Badakhshan, J. Geyer-Klingeberg, M. El-Halaby, T. Lutzeyer, and G.V.L. Affonseca. "Celonis Process Repository: A Bridge between Business Process Management and Process Mining". In: *Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track at BPM 2020 co-located with the 18th International Conference on Business Process Management (BPM 2020), Sevilla, Spain, September 13-18, 2020*. Vol. 2673. CEUR Workshop Proceedings. CEUR-WS.org, 2020, pp. 67–71. URL: http://ceur-ws.org/Vol-2673/paperDR01.pdf.

[15] E.L. Barse, H. Kvarnström, and E. Jonsson. “Synthesizing Test Data for Fraud Detection Systems”. In: *19th Annual Computer Security Applications Conference (ACSAC 2003), 8-12 December 2003, Las Vegas, NV, USA*. IEEE Computer Society, 2003, pp. 384–394. DOI: 10.1109/CSAC.2003.1254343. URL: https://doi.org/10.1109/CSAC.2003.1254343.

[16] I. Beerepoot, X. Lu, I. van de Weerd, and H.A. Reijers. “Seeing the Signs of Workarounds: A Mixed-Methods Approach to the Detection of Nurses’ Process Deviations”. In: (2021), pp. 1–10. URL: https://hdl.handle.net/10125/71072.

[17] I. Beerepoot, A. Ouali, I. van de Weerd, and H.A. Reijers. “Working around Health Information Systems: to Accept or not to Accept?” In: (2019). URL: https://aisel.aisnet.org/ecis2019_rp/182.

[18] K. Böhmer and S. Rinderle-Ma. “Multi-perspective Anomaly Detection in Business Process Execution Events”. In: *On the Move to Meaningful Internet Systems: OTM 2016 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2016, Rhodes, Greece, October 24-28, 2016, Proceedings*. Vol. 10033. Lecture Notes in Computer Science. 2016, pp. 80–98. DOI: 10.1007/978-3-319-48472-3_5. URL: https://doi.org/10.1007/978-3-319-48472-3_5.

[19] M.C. Boudreau and D. Robey. “Enacting Integrated Information Technology: A Human Agency Perspective”. In: *Organ. Sci.* 16.1 (2005), pp. 3–18. DOI: 10.1287/orsc.1040.0103. URL: https://doi.org/10.1287/orsc.1040.0103.

[20] Z.D. Bozorgi, I. Teinemaa, M. Dumas, M. La Rosa, and A. Polyvyanyy. “Process Mining Meets Causal Machine Learning: Discovering Causal Rules from Event Logs”. In: *2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, October 4-9, 2020*. IEEE, 2020, pp. 129–136. DOI: 10.1109/ICPM49681.2020.00028. URL: https://doi.org/10.1109/ICPM49681.2020.00028.

[21] F. Cabitza and C. Simone. “”Drops Hollowing the Stone”: Workarounds as Resources for Better Task-Artifact Fit”. In: *ECSCW 2013: Proceedings of the 13th European Conference on Computer Supported Cooperative Work, 21-25 September 2013, Paphos, Cyprus*. Springer, 2013, pp. 103–122. DOI: 10.1007/978-1-4471-5346-7_6. URL: https://doi.org/10.1007/978-1-4471-5346-7_6.

[22] R. Chalapathy and S. Chawla. “Deep Learning for Anomaly Detection: A Survey”. In: *CoRR* abs/1901.03407 (2019). arXiv: 1901.03407. URL: http://arxiv.org/abs/1901.03407.

[23] V. Chandola, A. Banerjee, and V. Kumar. "Anomaly detection: A survey". In: *ACM Comput. Surv.* 41.3 (2009), 15:1–15:58. DOI: 10.1145/1541880.1541882. URL: https://doi.org/10.1145/1541880.1541882.

[24] L. Sammani Chandradeva, T. Madushanka Amarasinghe, M. De Silva, A. Chathuranga Aponso, and N. Krishnarajah. "Monetary Transaction Fraud Detection System Based on Machine Learning Strategies". In: *Fourth International Congress on Information and Communication Technology - ICICT 2019, London, UK, February 25-26, 2019, Volume 1*. Vol. 1041. Advances in Intelligent Systems and Computing. Springer, 2019, pp. 385–396. DOI: 10.1007/978-981-15-0637-6_33. URL: https://doi.org/10.1007/978-981-15-0637-6_33.

[25] L. Copeland, D. Edberg, A.K. Panorska, and J. Wendel. "Applying business intelligence concepts to Medicaid claim fraud detection". In: *Journal of Information Systems Applied Research* 5.1 (2012), p. 51.

[26] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A.A. Bharath. "Generative Adversarial Networks: An Overview". In: *IEEE Signal Process. Mag.* 35.1 (2018), pp. 53–65. DOI: 10.1109/MSP.2017.2765202. URL: https://doi.org/10.1109/MSP.2017.2765202.

[27] M. Cui, J. Wang, and M. Yue. "Machine Learning-Based Anomaly Detection for Load Forecasting Under Cyberattacks". In: *IEEE Trans. Smart Grid* 10.5 (2019), pp. 5724–5734. DOI: 10.1109/TSG.2018.2890809. URL: https://doi.org/10.1109/TSG.2018.2890809.

[28] J. Vieira da Cunha and A. Carugati. "Information technology and the first-line manager's dilemma: Lessons from an ethnographic study". In: *17th European Conference on Information Systems, ECIS 2009, Verona, Italy, 2009*. 2009, pp. 2834–2845. URL: http://aisel.aisnet.org/ecis2009/323.

[29] A.K.A De Medeiros and C.W. Günther. "Process mining: Using CPN tools to create test logs for mining algorithms". In: *Proceedings of the sixth workshop on the practical use of coloured Petri nets and CPN tools (CPN 2005)*. Vol. 576. Citeseer. 2005.

[30] E. Duman and M.H. Özçelik. "Detecting credit card fraud by genetic algorithm and scatter search". In: *Expert Syst. Appl.* 38.10 (2011), pp. 13057–13063. DOI: 10.1016/j.eswa.2011.04.110. URL: https://doi.org/10.1016/j.eswa.2011.04.110.

[31] C. Esteban, S.L. Hyland, and G. Rätsch. "Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs". In: *CoRR* abs/ 1706.02633 (2017). arXiv: 1706.02633. URL: http://arxiv.org/abs/1706.02633.

[32] D. Fernández-Cerero, A. Fernández-Montes, A. Jakobik, J. Kolodziej, and M. Toro. “SCORE: Simulator for cloud optimization of resources and energy consumption”. In: *Simul. Model. Pract. Theory* 82 (2018), pp. 160–173. DOI: 10.1016/j.simpat.2018.01.004. URL: https://doi.org/10.1016/j.simpat.2018.01.004.

[33] E. Ferrara, O. Varol, C.A. Davis, F. Menczer, and A. Flammini. “The rise of social bots”. In: *Commun. ACM* 59.7 (2016), pp. 96–104. DOI: 10.1145/2818717. URL: http://doi.acm.org/10.1145/2818717.

[34] P.A. Gagniuc. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

[35] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B.Y. Zhao. “Detecting and characterizing social spam campaigns”. In: *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*. ACM, 2010, pp. 681–683. DOI: 10.1145/1866307.1866396. URL: https://doi.org/10.1145/1866307.1866396.

[36] Z. Gilani, E. Kochmar, and J. Crowcroft. “Classification of Twitter Accounts into Automated Agents and Human Users”. In: *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, Sydney, Australia, July 31 - August 03, 2017*. ACM, 2017, pp. 489–496. DOI: 10.1145/3110025.3110091. URL: https://doi.org/10.1145/3110025.3110091.

[37] M. Goldstein and S. Uchida. “A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data”. In: *PloS one* 11.4 (2016), e0152173.

[38] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A.C. Courville, and Y. Bengio. “Generative Adversarial Nets”. In: (2014), pp. 2672–2680.

[39] N. Görnitz, M. Kloft, K. Rieck, and U. Brefeld. “Toward Supervised Anomaly Detection”. In: *CoRR* abs/1401.6424 (2014). arXiv: 1401.6424. URL: http://arxiv.org/abs/1401.6424.

[40] F.E. Grubbs. “Procedures for detecting outlying observations in samples”. In: *Technometrics* 11.1 (1969), pp. 1–21.

[41] X. Guo, Y. Yin, C. Dong, G. Yang, and G. Zhou. “On the Class Imbalance Problem”. In: *Fourth International Conference on Natural Computation, ICNC 2008, Jinan, Shandong, China, 18-20 October 2008, Volume 4*. IEEE Computer Society, 2008, pp. 192–201. DOI: 10.1109/ICNC.2008.871. URL: https://doi.org/10.1109/ICNC.2008.871.

[42] J.R.B. Halbesleben and C. Rathert. “The role of continuous quality improvement and psychological safety in predicting work-arounds”. In: *Health care management review* 33.2 (2008), pp. 134–144.

[43] HH.aibo He and E.A. Garcia. “Learning from Imbalanced Data”. In: *IEEE Trans. Knowl. Data Eng.* 21.9 (2009), pp. 1263–1284. DOI: `10.1109/TKDE.2008.239`. URL: `https://doi.org/10.1109/TKDE.2008.239`.

[44] P.I. Hofgesang and W. Kowalczyk. “Analysing clickstream data: From anomaly detection to visitor profiling”. In: *Proc. of ECML/PKDD Discovery Challenge* (2005).

[45] M. Hosseinpour and M. Jans. “Categorizing Identified Deviations for Auditing”. In: *Proceedings of the 6th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2016), Graz, Austria, 2016*. Vol. 1757. CEUR Workshop Proceedings. CEUR-WS.org, 2016, pp. 125–129.

[46] S. Huda, R. Sarno, and T. Ahmad. “Fuzzy MADM approach for Rating of Process-based Fraud”. In: *Journal of ICT Research and Applications* 9.2 (2015), pp. 111–128.

[47] L. Igual and S. Seguí. “Introduction to Data Science - A Python Approach to Concepts, Techniques and Applications”. In: Undergraduate Topics in Computer Science. Springer, 2017. ISBN: 978-3-319-50016-4. DOI: `10.1007/978-3-319-50017-1`. URL: `https://doi.org/10.1007/978-3-319-50017-1`.

[48] *ING betaalt 775 miljoen vanwege ernstige nalatigheden bij voorkomen witwassen*. Sept. 4, 2018. URL: `https://www.fiod.nl/ing-betaalt-775-miljoen-vanwege-ernstige-nalatigheden-bij-voorkomen-witwassen/` (visited on 07/10/2022).

[49] G. Jacob, E. Kirda, C. Kruegel, and G. Vigna. “PUBCRAWL: Protecting Users and Businesses from CRAWLers”. In: *Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012*. USENIX Association, 2012, pp. 507–522.

[50] G. Jacobusse and C.J. Veenman. “On Selection Bias with Imbalanced Classes”. In: *Discovery Science - 19th International Conference, DS 2016, Bari, Italy, October 19-21, 2016, Proceedings*. Vol. 9956. Lecture Notes in Computer Science. 2016, pp. 325–340. DOI: `10.1007/978-3-319-46307-0\_21`. URL: `https://doi.org/10.1007/978-3-319-46307-0\_21`.

[51] S.N. John, C. Anele, O.O Kennedy, F. Olajide, and C.G. Kennedy. “Realtime fraud detection in the banking sector using data mining techniques/algorithm”. In: *2016 international conference on computational science and computational intelligence (CSCI)*. IEEE. 2016, pp. 1186–1191.

[52] J. Jurgovsky, M. Granitzer, K. Ziegler, S. Calabretto, P.-Edouard Portier, L. He-Guelton, and O. Caelen. “Sequence classification for credit-card fraud detection”. In: *Expert Syst. Appl.* 100 (2018), pp. 234–245. DOI: 10.1016/j.eswa.2018.01.037. URL: https://doi.org/10.1016/j.eswa.2018.01.037.

[53] D. Kenyon and J.H.P. Eloff. “Big data science for predicting insurance claims fraud”. In: *2017 Information Security for South Africa, ISSA 2017, Johannesburg, South Africa, August 16-17, 2017*. IEEE, 2017, pp. 40–47. DOI: 10.1109/ISSA.2017.8251773. URL: https://doi.org/10.1109/ISSA.2017.8251773.

[54] J. Ko and M. Comuzzi. “Detecting anomalies in business process event logs using statistical leverage”. In: *Inf. Sci.* 549 (2021), pp. 53–67. DOI: 10.1016/j.ins.2020.11.017. URL: https://doi.org/10.1016/j.ins.2020.11.017.

[55] L. Kocsis and A. György. “Fraud detection by generating positive samples for classification from unlabeled data”. In: *ICML 2010 Workshop on Machine Learning and Games*. 2010.

[56] S. Kovach and W.V. Ruggiero. “Online banking fraud detection based on local and global behavior”. In: *Proc. of the Fifth International Conference on Digital Society, Guadeloupe, France*. 2011, pp. 166–171.

[57] S. Kudugunta and E. Ferrara. “Deep neural networks for bot detection”. In: *Inf. Sci.* 467 (2018), pp. 312–322. DOI: 10.1016/j.ins.2018.08.019. URL: https://doi.org/10.1016/j.ins.2018.08.019.

[58] M. Kumar, R. Ghani, and Z. Mei. “Data mining to predict and prevent errors in health insurance claims processing”. In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010*. ACM, 2010, pp. 65–74. DOI: 10.1145/1835804.1835816. URL: https://doi.org/10.1145/1835804.1835816.

[59] S. Kumar, F. Spezzano, and V.S. Subrahmanian. “VEWS: A Wikipedia Vandal Early Warning System”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. ACM, 2015, pp. 607–616. DOI: 10.1145/2783258.2783367. URL: https://doi.org/10.1145/2783258.2783367.

[60] L. Lapointe and S. Rivard. "A Multilevel Model of Resistance to Information Technology Implementation". In: *MIS Q.* 29.3 (2005), pp. 461–491. URL: http://misq.org/a-multilevel-model-of-resistance-to-information-technology-implementation.html.

[61] S.K. Lim, Y. Loo, N. Tran, N. Cheung, G. Roig, and Y. Elovici. "DOPING: Generative Data Augmentation for Unsupervised Anomaly Detection with GAN". In: *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*. IEEE Computer Society, 2018, pp. 1122–1127. DOI: 10.1109/ICDM.2018.00146. URL: https://doi.org/10.1109/ICDM.2018.00146.

[62] F. de Lima Bezerra, J. Wainer, and W.M.P. van der Aalst. "Anomaly Detection Using Process Mining". In: *Enterprise, Business-Process and Information Systems Modeling, 10th International Workshop, BPMDS 2009, and 14th International Conference, EMMSAD 2009, held at CAiSE 2009, Amsterdam, The Netherlands, June 8-9, 2009. Proceedings*. Vol. 29. Lecture Notes in Business Information Processing. Springer, 2009, pp. 149–161. DOI: 10.1007/978-3-642-01862-6_13. URL: https://doi.org/10.1007/978-3-642-01862-6_13.

[63] W. Lin, L. Sun, Q. Zhong, C. Liu, J. Feng, X. Ao, and H. Yang. "Online Credit Payment Fraud Detection via Structure-Aware Hierarchical Recurrent Neural Network". In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*. ijcai.org, 2021, pp. 3670–3676. DOI: 10.24963/ijcai.2021/505. URL: https://doi.org/10.24963/ijcai.2021/505.

[64] F.T. Liu, K.M. Ting, and Z. Zhou. "Isolation forest". In: *2008 eighth ieee international conference on data mining*. IEEE. 2008, pp. 413–422.

[65] M.M. Masud, C. Woolam, J. Gao, L. Khan, J. Han, K.W. Hamlen, and N.C. Oza. "Facing the reality of data stream classification: coping with scarcity of labeled data". In: *Knowl. Inf. Syst.* 33.1 (2011), pp. 213–244. DOI: 10.1007/s10115-011-0447-8. URL: https://doi.org/10.1007/s10115-011-0447-8.

[66] A.C. Müller and S. Guido. *Introduction to machine learning with Python: a guide for data scientists.* " O'Reilly Media, Inc.", 2016, pp. 72–85.

[67] T. Nolle, S. Luettgen, A. Seeliger, and M. Mühlhäuser. "Analyzing business process anomalies using autoencoders". In: *Mach. Learn.* 107.11 (2018), pp. 1875–1893. DOI: 10.1007/s10994-018-5702-8. URL: https://doi.org/10.1007/s10994-018-5702-8.

[68] T. Nolle, S. Luettgen, A. Seeliger, and M. Mühlhäuser. “BINet: Multi-perspective business process anomaly classification”. In: *Inf. Syst.* 103 (2022), p. 101458. DOI: 10.1016/j.is.2019.101458. URL: https://doi.org/10.1016/j.is.2019.101458.

[69] B. Omair and A. Alturki. “A Systematic Literature Review of Fraud Detection Metrics in Business Processes”. In: *IEEE Access* 8 (2020), pp. 26893–26903. DOI: 10.1109/ACCESS.2020.2971604. URL: https://doi.org/10.1109/ACCESS.2020.2971604.

[70] E.S. Pane, A.D. Wibawa, and M.H. Purnomo. “Event log-based fraud rating using interval type-2 fuzzy sets in fuzzy AHP”. In: *2016 IEEE region 10 conference (TENCON)*. IEEE. 2016, pp. 1965–1968.

[71] G. Pang, C. Shen, L. Cao, and A. van den Hengel. “Deep Learning for Anomaly Detection: A Review”. In: *ACM Comput. Surv.* 54.2 (2021), 38:1–38:38. DOI: 10.1145/3439950. URL: https://doi.org/10.1145/3439950.

[72] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee. “A Design Science Research Methodology for Information Systems Research”. In: *J. Manag. Inf. Syst.* 24.3 (2008), pp. 45–77. DOI: 10.2753/mis0742-1222240302. URL: https://doi.org/10.2753/mis0742-1222240302.

[73] M. Pegoraro and W.M.P. van der Aalst. “Mining Uncertain Event Data in Process Mining”. In: *International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24-26, 2019*. IEEE, 2019, pp. 89–96. DOI: 10.1109/ICPM.2019.00023. URL: https://doi.org/10.1109/ICPM.2019.00023.

[74] J. Pérez, P. Arroba, and J.é Manuel Moya. “Data augmentation through multivariate scenario forecasting in Data Centers using Generative Adversarial Networks”. In: *CoRR* abs/2201.06147 (2022). arXiv: 2201.06147. URL: https://arxiv.org/abs/2201.06147.

[75] C. Phua, V. C. S. Lee, K. Smith-Miles, and R. W. Gayler. “A Comprehensive Survey of Data Mining-based Fraud Detection Research”. In: *CoRR* abs/1009.6119 (2010). arXiv: 1009.6119. URL: http://arxiv.org/abs/1009.6119.

[76] M. Pourbafrani, S. Vasudevan, F. Zafar, Y. Xingran, R. Singh, and W.M.P. van der Aalst. “A Python Extension to Simulate Petri nets in Process Mining”. In: *CoRR* abs/2102.08774 (2021). arXiv: 2102.08774. URL: https://arxiv.org/abs/2102.08774.

[77] D. Rahmawati, R. Sarno, C. Fatichah, and D. Sunaryono. “Fraud detection on event log of bank financial credit business process using Hidden Markov Model algorithm”. In: *2017 3rd International Conference on Science in Information Technology (ICSITech)*. IEEE. 2017, pp. 35–40.

[78] L. Reinkemeyer. “Process mining in action”. In: *Process Mining in Action Principles, Use Cases and Outloook* (2020).

[79] L. Ruff, J.R. Kauffmann, R.A. Vandermeulen, G. Montavon, W. Samek, Ma. Kloft, T.G. Dietterich, and K.R. Müller. “A Unifying Review of Deep and Shallow Anomaly Detection”. In: *Proc. IEEE* 109.5 (2021), pp. 756–795. DOI: 10.1109/JPROC.2021.3052449. URL: https://doi.org/10.1109/JPROC.2021.3052449.

[80] A. Rzad, J. Wojnecka, M. Rutkowski, and M. Guliński. “Investigating Purchase-to-Pay process using Process Mining in a multinational corporation”. In: (2019).

[81] N. Sadagopan and J. Li. “Characterizing typical and atypical user sessions in clickstreams”. In: *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*. ACM, 2008, pp. 885–894. DOI: 10.1145/1367497.1367617. URL: https://doi.org/10.1145/1367497.1367617.

[82] M. Sahlabadi, R. Chandren Muniyandi, and Z. Shukur. “Detecting Abnormal Behavior in Social Network Websites by using a Process Mining Technique”. In: *J. Comput. Sci.* 10.3 (2014), pp. 393–402. DOI: 10.3844/jcssp.2014.393.402. URL: https://doi.org/10.3844/jcssp.2014.393.402.

[83] M.S. Santos, J.P. Soares, P.H. Abreu, H.r Araújo, and J.A.M. Santos. “Cross-Validation for Imbalanced Datasets: Avoiding Overoptimistic and Overfitting Approaches”. In: *IEEE Comput. Intell. Mag.* 13.4 (2018), pp. 59–76. DOI: 10.1109/MCI.2018.2866730. URL: https://doi.org/10.1109/MCI.2018.2866730.

[84] R. Sarno, R.D Dustrial Dewandono, T. Ahmad, M.F Naufal, and F. Sinaga. “Hybrid Association Rule Learning and Process Mining for Fraud Detection.” In: *IAENG International Journal of Computer Science* 42.2 (2015).

[85] R. Sarno, F. Sinaga, and K.R Sungkono. “Anomaly detection in business processes using process mining and fuzzy association rule learning”. In: *J. Big Data* 7.1 (2020), p. 5. DOI: 10.1186/s40537-019-0277-1. URL: https://doi.org/10.1186/s40537-019-0277-1.

[86] J. Sun, Y. Li, C. Chen, J. Lee, X. Liu, Z. Zhang, L. Huang, L. Shi, and W. Xu. "FDHelper: Assist Unsupervised Fraud Detection Experts with Interactive Feature Selection and Evaluation". In: *CHI '20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020*. ACM, 2020, pp. 1–12. DOI: 10.1145/3313831.3376140. URL: https://doi.org/10.1145/3313831.3376140.

[87] J. Sun, Q. Zhu, Z. Liu, X. Liu, J. Lee, Z. Su, L. Shi, L. Huang, and W. Xu. "Fraud-Vis: Understanding Unsupervised Fraud Detection Algorithms". In: *IEEE Pacific Visualization Symposium, PacificVis 2018, Kobe, Japan, April 10-13, 2018*. IEEE Computer Society, 2018, pp. 170–174. DOI: 10.1109/PacificVis.2018.00029. URL: https://doi.org/10.1109/PacificVis.2018.00029.

[88] N. Tax, K.J de Vries, M. de Jong, N. Dosoula, B. van den Akker, J. Smith, O. Thuong, and L. Bernardi. "Machine Learning for Fraud Detection in E-Commerce: A Research Agenda". In: *CoRR* abs/2107.01979 (2021). arXiv: 2107.01979. URL: https://arxiv.org/abs/2107.01979.

[89] T. Taewoong Um, F.M.J Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulic. "Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks". In: *Proceedings of the 19th ACM International Conference on Multimodal Interaction, ICMI 2017, Glasgow, United Kingdom, November 13 - 17, 2017*. ACM, 2017, pp. 216–220. DOI: 10.1145/3136755.3136817. URL: https://doi.org/10.1145/3136755.3136817.

[90] O. Varol, E. Ferrara, C.A. Davis, F. Menczer, and A. Flammini. "Online Human-Bot Interactions: Detection, Estimation, and Characterization". In: *Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017*. AAAI Press, 2017, pp. 280–289. URL: https://aaai.org/ocs/index.php/ICWSM/ICWSM17/paper/view/15587.

[91] B. Viswanath, M. Ahmad Bashir, M. Crovella, S. Guha, K. P. Gummadi, B. Krishnamurthy, and A. Mislove. "Towards Detecting Anomalous User Behavior in Online Social Networks". In: *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*. USENIX Association, 2014, pp. 223–238. URL: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/viswanath.

[92] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B.Y. Zhao. "You Are How You Click: Clickstream Analysis for Sybil Detection". In: *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*. USENIX Association, 2013, pp. 241–256. URL: `https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/wang`.

[93] S. Wang, C. Liu, X. Gao, H. Qu, and W. Xu. "Session-Based Fraud Detection in Online E-Commerce Transactions Using Recurrent Neural Networks". In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18-22, 2017, Proceedings, Part III*. Vol. 10536. Lecture Notes in Computer Science. Springer, 2017, pp. 241–252. DOI: `10.1007/978-3-319-71273-4\_20`. URL: `https://doi.org/10.1007/978-3-319-71273-4\_20`.

[94] I. van de Weerd, P. Vollers, I. Beerepoot, and M. Fantinato. "Workarounds in Retail Work Systems: Prevent, redesign, Adopt or Ignore?" In: *27th European Conference on Information Systems - Information Systems for a Sharing Society, ECIS 2019, Stockholm and Uppsala, Sweden, June 8-14, 2019*. 2019. URL: `https://aisel.aisnet.org/ecis2019\_rp/183`.

[95] S. Weinzierl, V. Wolf, T. Pauli, D. Beverungen, and M. Matzner. "Detecting in Business Processes a Deep Learning method for Analyzing Event Logs". In: *28th European Conference on Information Systems - Liberty, Equality, and Fraternity in a Digitizing World, ECIS 2020, Marrakech, Morocco, June 15-17, 2020*. 2020. URL: `https://aisel.aisnet.org/ecis2020\_rp/67`.

[96] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, and B. Regnell. *Experimentation in Software Engineering*. Springer, 2012. ISBN: 978-3-642-29043-5. DOI: `10.1007/978-3-642-29044-2`. URL: `https://doi.org/10.1007/978-3-642-29044-2`.

[97] D. Xi, B. Song, F. Zhuang, Y. Zhu, S. Chen, T. Zhang, Y. Qi, and Q. He. "Modeling the Field Value Variations and Field Interactions Simultaneously for Fraud Detection". In: *CoRR* abs/2008.05600 (2020). arXiv: `2008.05600`. URL: `https://arxiv.org/abs/2008.05600`.

[98] D. Xi, F. Zhuang, B. Song, Y. Zhu, S. Chen, D. Hong, T. Chen, X. Gu, and Q. He. "Neural Hierarchical Factorization Machines for User's Event Sequence Analysis". In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July*

25-30, 2020. ACM, 2020, pp. 1893–1896. DOI: 10.1145/3397271.3401307. URL: https://doi.org/10.1145/3397271.3401307.

[99] Y. Yannikos, F. Franke, C. Winter, and M. Schneider. "3LSPG: Forensic Tool Evaluation by Three Layer Stochastic Process-Based Generation of Data". In: *Computational Forensics - 4th International Workshop, IWCF 2010, Tokyo, Japan, November 11-12, 2010, Revised Selected Papers*. Vol. 6540. Lecture Notes in Computer Science. Springer, 2010, pp. 200–211. DOI: 10.1007/978-3-642-19376-7_18. URL: https://doi.org/10.1007/978-3-642-19376-7_18.

[100] Jiawen Yu, Yiqiang Jiang, and Yanqiu Yan. "A simulation study on heat recovery of data center: A case study in Harbin, China". In: *Renewable energy* 130 (2019), pp. 154–173.

[101] G. Zhang, Z. Li, J. Huang, J. Wu, C. Zhou, J. Yang, and J. Gao. "eFraudCom: An E-commerce Fraud Detection System via Competitive Graph Neural Networks". In: *ACM Trans. Inf. Syst.* 40.3 (2022), 47:1–47:29. DOI: 10.1145/3474379. URL: https://doi.org/10.1145/3474379.

[102] R. Zhang, F. Zheng, and W. Min. "Sequential Behavioral Data Processing Using Deep Learning and the Markov Transition Field in Online Fraud Detection". In: *CoRR* abs/1808.05329 (2018). arXiv: 1808.05329. URL: http://arxiv.org/abs/1808.05329.

[103] Z. Zhang, L. Chen, Q. Liu, and P. Wang. "A Fraud Detection Method for Low-Frequency Transaction". In: *IEEE Access* 8 (2020), pp. 25210–25220. DOI: 10.1109/ACCESS.2020.2970614. URL: https://doi.org/10.1109/ACCESS.2020.2970614.

[104] L. Zheng, G. Liu, W. Luan, Z. Li, Y. Zhang, C. Yan, and C. Jiang. "A new credit card fraud detecting method based on behavior certificate". In: *15th IEEE International Conference on Networking, Sensing and Control, ICNSC 2018, Zhuhai, China, March 27-29, 2018*. IEEE, 2018, pp. 1–6. DOI: 10.1109/ICNSC.2018.8361328. URL: https://doi.org/10.1109/ICNSC.2018.8361328.

[105] X. Zhu, X. Ao, Z. Qin, Y. Chang, Y. Liu, Q. He, and J. Li. "Intelligent financial fraud detection practices in post-pandemic era". In: *The Innovation* 2.4 (2021), p. 100176.

[106] Y. Zhu, D. Xi, B. Song, F. Zhuang, S. Chen, X. Gu, and Q. He. "Modeling Users' Behavior Sequences with Hierarchical Explainable Network for Cross-domain Fraud Detection". In: *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*. ACM / IW3C2, 2020, pp. 928–938. DOI: 10.1145/3366423.3380172. URL: https://doi.org/10.1145/3366423.3380172.

A Taxonomy

Characteristic	**Reference**
Skipped event	[51, 45, 85, 84, 68, 67, 69, 77, 46, 70, 54]
Inserted event	[45, 67, 54]
Loop on event	[45, 68]
Repetition of an event	[95, 45, 67]
Swapping two events	[45, 68]
Replacing one event by another	[45, 67]
Wrong roles / wrong resources	[51, 85, 84, 77, 46, 70]
Wrong attributes	[51, 85, 67]
Wrong throughput time	[51, 84, 85, 69, 77, 46, 70]
Wrong duty	[51, 85, 67, 69, 77, 46, 70]
Wrong decision	[51, 85, 69, 77, 46, 70]
Unexpected events	[18]
Collective event anomalies	[18]
Rework	[14, 80, 70]
Workarounds	[16, 95]
Parallel	[69, 46]

Table 23: Overview of process mining anomalies

B Correspondence with UWV

B.1 Reaching out to the UWV domain expert

B.1.1 Original first correspondence

Dear [],

My name is Jochem Veldman and I am a master student of Business Informatics at Utrecht University. Currently, I am working on my master thesis and I am getting guidance from Xixi Lu. Xixi gave me your name because for my thesis research we want to use the UWV dataset from the BPI Challenge 2016.

For my thesis, I am researching how generated anomaly/fraud behaviour can be added to an event log containing ”normal” behaviour. We want to generate the anomaly behaviour using characteristics described in the literature. For this generation we want to use process mining. Currently, the scope of the research is focused on user behaviour in an online environment. The UWV dataset of 2016 is one of the few process mining datasets available online in this field and therefore we would like to use it.

I would like to schedule a short Teams meeting with you to discuss which anomalies you expect to possibly be in the UWV dataset and which are interesting to detect. Based on this information, I will then know which anomalies I can generate and detect in the dataset.

I can at the following times:x

List of possible data

I hope that there is a moment in between that is convenient for you.

Kind regards,

Jochem Veldman

B.1.2 Second correspondence

Dear [],

A while ago we sat down together and discussed that I would come back to you around 1 April to let you know what the next plan of action for my thesis is.

I have used the past time to better understand the UWV dataset. I now have a better idea of what variables and activities there are and how this can fit in with my thesis topic. Looking at most characteristics of process mining anomalies is difficult because there is no defined process underlying the data. Therefore, Xixi and I figured out that it is most interesting and relevant (for now) to look at rehearsals in the sessions. For rehearsals, it is less necessary/less necessary to know the underlying process. We would like to hear what you think about this.

In the coming period we want to look further into how to generate the anomalies with the rehearsal characteristic. Then we want to add the generated anomalies to the dataset and see what the effect of this is on the model and the anomalies that the model detects.

Have a nice weekend!

Kind regards,

Jochem Veldman

B.1.3 Second correspondence response

Hi Jochem,

Good to hear that you still see a point of reference in the dataset.

Looking at repetitions is interesting in several ways.

1. Within one session it can indicate a part that is not clear to a customer.
2. Across all sessions of one client it can indicate compliance with the application requirements. Rehearsals can therefore be both regular and striking.
3. The absence of repetitions can also be striking.
   (a) Certain repetitions should occur in all sessions of one client. For example, related to the 4-weekly reporting of the application activities. If this rehearsal is missing, there might be something wrong.
   (b) Within 1 session I expect that certain repetitions can occur regularly. For example, because a customer has the same page several times in a row in the session, which is because the data cannot always show exactly what a customer is doing on a page. If this repetition is missing, there might have been something wrong. Perhaps a technical problem.
   (c) On the other hand, within 1 session, striking repetitions can also occur (so not regularly expected). In this case, I expect that especially the return to a previously visited page, with other visited pages in between, can be an indication that something is not clear to a customer.
4. Finally, I expect that there will also be different types of revisits depending on how long a customer has been at the UWV. In the beginning, customers often visit slightly different pages (for different reasons) than after a few months.

   In any case, I am curious and interested in your findings. As soon as you have something to show, let me know and we can make an appointment. If you have any questions before then, please let me know as well.

Regards,
[]

C Data distribution visualisation

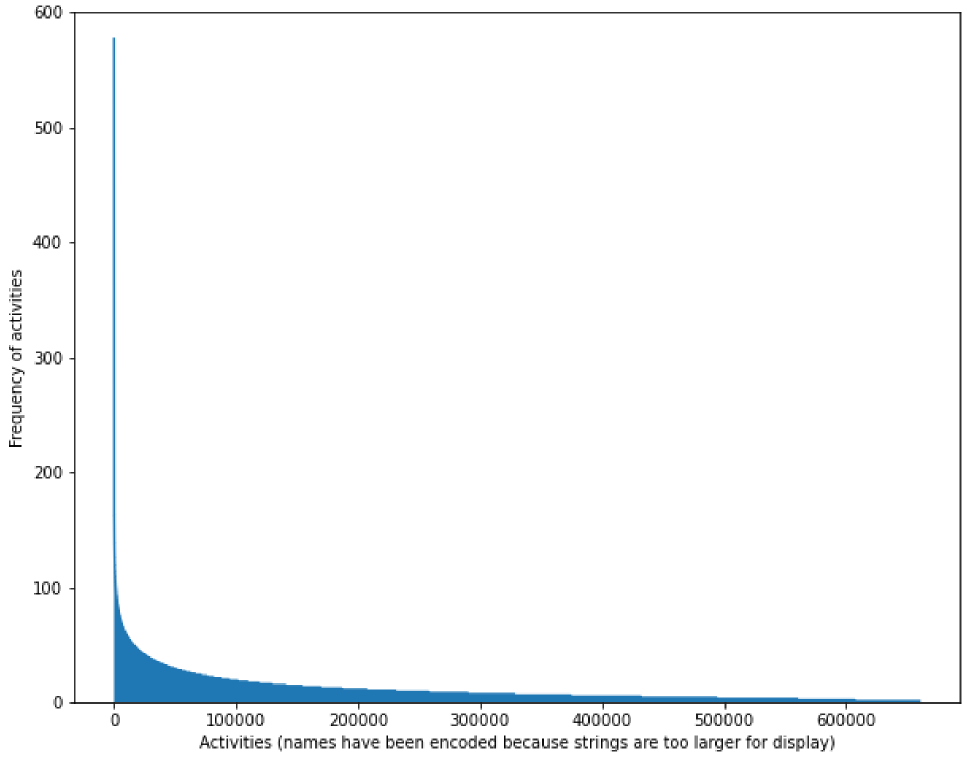


Figure 34: Distribution of activities per session

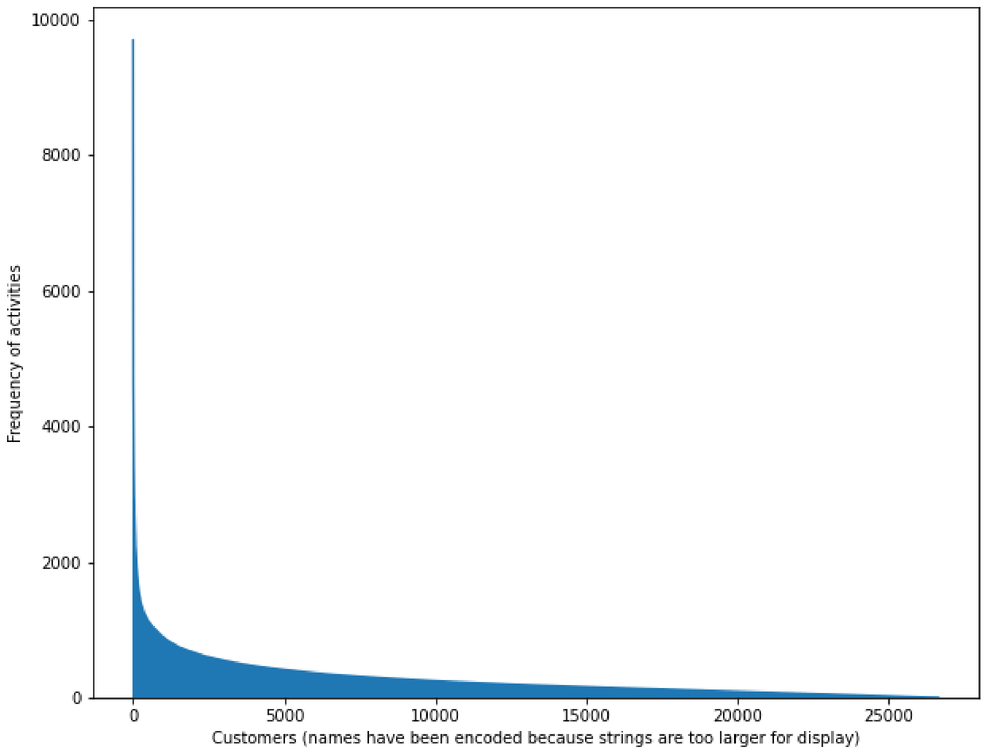


Figure 35: Distribution of activities per customer

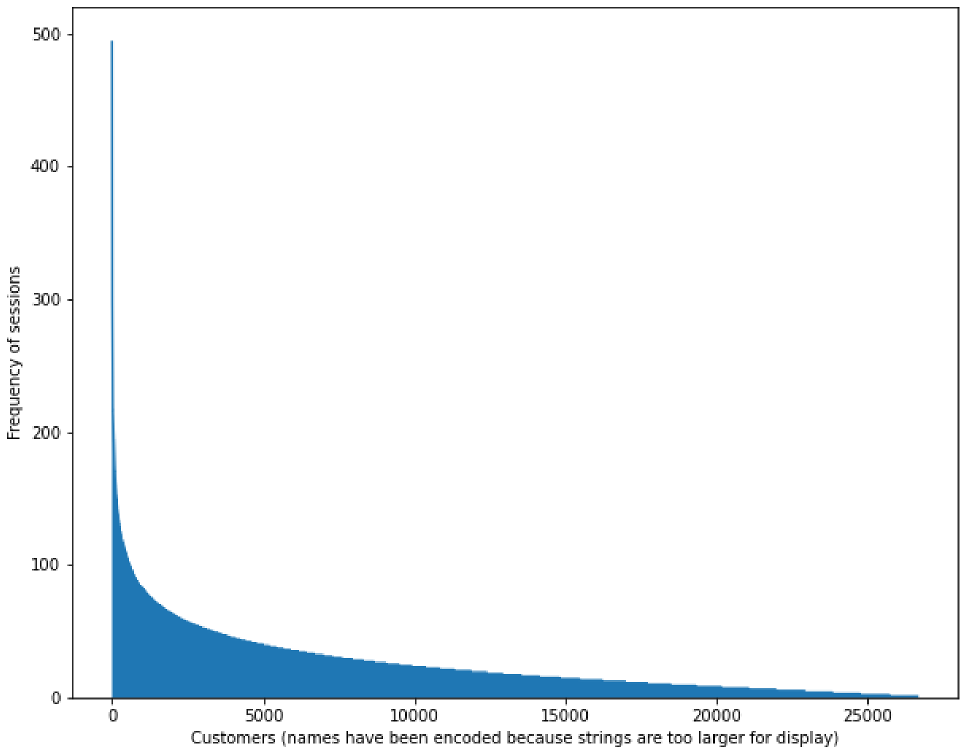


Figure 36: Distribution of performed session frequency per customer

D Tables AUC scores

Number of anomalous sessions injected	AUC score probability min 0.55	AUC score probability plus 0	AUC score probability plus 0.2	AUC score probability plus 1	AUC score probability plus 3	AUC score probability plus 5	AUC score probability plus 10	AUC score probability plus 25
0	0.666	0.666	0.666	0.666	0.666	0.666	0.666	0.666
10	0.617	0.662	0.649	0.680	0.681	0.698	0.671	0.670
25	0.609	0.693	0.690	0.732	0.741	0.749	0.671	0.708
50	0.598	0.751	0.747	0.799	0.817	0.797	0.697	0.713
75	0.586	0.759	0.779	0.847	0.855	0.799	0.705	0.738
100	0.591	0.796	0.831	0.859	0.860	0.832	0.730	0.760
200	0.636	0.931	0.946	0.942	0.912	0.905	0.813	0.841
300	0.656	0.969	0.982	0.967	0.934	0.925	0.817	0.847
400	0.687	0.987	0.984	0.978	0.952	0.936	0.835	0.872
500	0.715	0.998	0.987	0.984	0.968	0.954	0.854	0.880
600	0.747	0.997	0.995	0.985	0.979	0.959	0.860	0.886
700	0.767	0.997	0.997	0.987	0.982	0.962	0.874	0.896
800	0.810	0.997	0.997	0.989	0.981	0.967	0.881	0.908
900	0.862	0.997	0.997	0.993	0.985	0.968	0.886	0.915
1000	0.885	0.995	0.997	0.993	0.986	0.978	0.897	0.917
1250	0.924	0.995	0.996	0.992	0.990	0.984	0.910	0.924
1500	0.956	0.994	0.995	0.994	0.991	0.990	0.915	0.929
2000	0.982	0.994	0.994	0.995	0.994	0.992	0.926	0.953

Table 24: AUC scores of models trained with different quantities of anomalies and different types of anomalies.

Number of anomalous sessions injected	AUC score probability min 0.1	AUC score probability plus 0	AUC score probability plus 0.2	AUC score probability plus 1	AUC score probability plus 3	AUC score probability plus 5	AUC score probability plus 10	AUC score probability plus 25
0	0.623	0.623	0.623	0.623	0.623	0.623	0.623	0.623
10	0.636	0.636	0.636	0.649	0.643	0.649	0.649	0.688
20	0.623	0.61	0.623	0.669	0.669	0.656	0.649	0.675
50	0.63	0.617	0.617	0.708	0.701	0.701	0.695	0.662
75	0.617	0.623	0.63	0.714	0.753	0.747	0.701	0.695
100	0.63	0.63	0.643	0.753	0.766	0.773	0.721	0.708
200	0.656	0.694	0.714	0.805	0.818	0.883	0.792	0.831
300	0.688	0.714	0.772	0.869	0.876	0.915	0.889	0.889
400	0.7	0.778	0.824	0.895	0.921	0.941	0.921	0.928
500	0.713	0.817	0.869	0.914	0.927	0.94	0.947	0.94
600	0.745	0.855	0.895	0.94	0.953	0.947	0.953	0.947
700	0.771	0.888	0.907	0.953	0.966	0.966	0.959	0.959
800	0.809	0.894	0.933	0.959	0.966	0.985	0.959	0.966
900	0.809	0.9	0.939	0.965	0.985	0.991	0.978	0.978
1000	0.815	0.907	0.946	0.972	0.985	0.998	0.985	0.985
1250	0.854	0.919	0.952	0.984	0.991	0.997	0.997	0.997
1500	0.886	0.938	0.957	0.984	0.997	0.997	0.996	0.996
2000	0.905	0.944	0.976	0.995	0.996	0.996	0.995	0.996

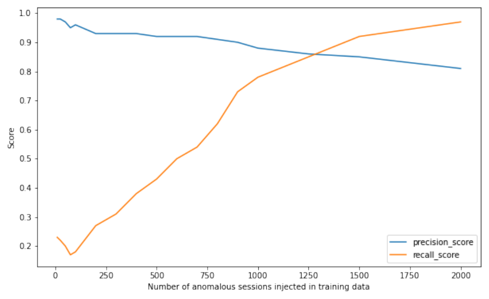
Table 25: AUC scores of models trained with different quantities of injected anomalous session and different types of anomalies for dirrect-follow pattern.

Number of anomalous sessions injected	AUC score probability plus 0	AUC score probability plus 0.2	AUC score probability plus 1	AUC score probability plus 3	AUC score probability plus 5	AUC score probability plus 10	AUC score probability plus 25
0	0.577	0.577	0.577	0.577	0.577	0.577	0.577
10	0.569	0.572	0.56	0.574	0.566	0.555	0.556
25	0.577	0.569	0.561	0.574	0.561	0.569	0.553
50	0.585	0.582	0.558	0.576	0.566	0.569	0.558
75	0.595	0.602	0.575	0.579	0.571	0.573	0.563
100	0.621	0.607	0.588	0.585	0.577	0.57	0.56
200	0.693	0.618	0.596	0.598	0.589	0.58	0.563
300	0.752	0.627	0.606	0.606	0.595	0.588	0.56
400	0.791	0.634	0.614	0.611	0.592	0.593	0.563
500	0.823	0.64	0.612	0.609	0.596	0.593	0.57
600	0.86	0.644	0.614	0.611	0.599	0.589	0.574
700	0.887	0.649	0.614	0.612	0.594	0.589	0.577
800	0.897	0.656	0.617	0.616	0.596	0.59	0.576
900	0.917	0.661	0.619	0.62	0.597	0.59	0.577
1000	0.936	0.67	0.62	0.619	0.597	0.591	0.58
1250	0.957	0.679	0.627	0.631	0.607	0.596	0.577
1500	0.968	0.692	0.633	0.636	0.606	0.6	0.574
2000	0.98	0.71	0.636	0.64	0.613	0.6	0.572

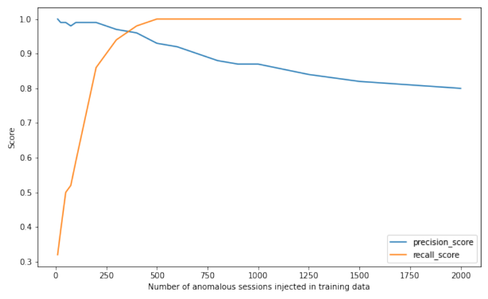
Table 26: AUC scores of models trained with different quantities of injected anomalous session and different types of anomalies for advanced repetition pattern.

E Precision recall plots

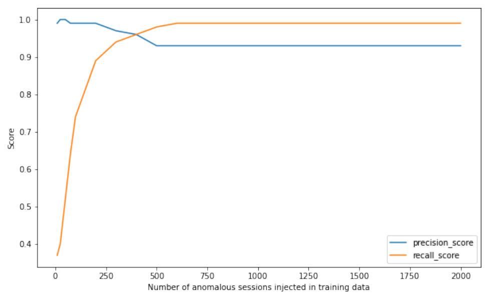
E.1 Precision recall plots for repetition pattern



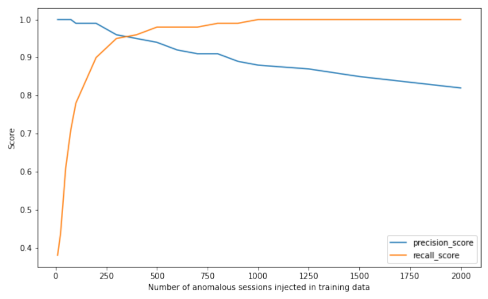
(a) Precision and recall scores for repetition pattern with min 0.55



(b) Precision and recall scores for repetition pattern with deviation rate = 0

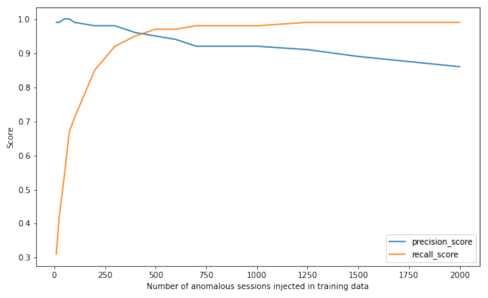


(c) Precision and recall scores for repetition pattern with deviation rate = 0.2

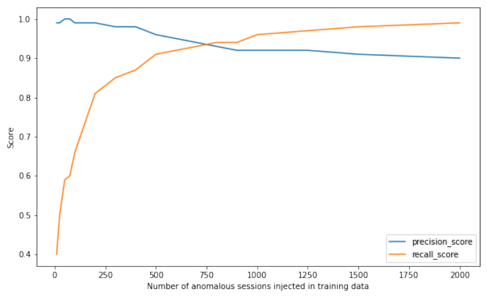


(d) Precision and recall scores for repetition pattern with deviation rate = 1

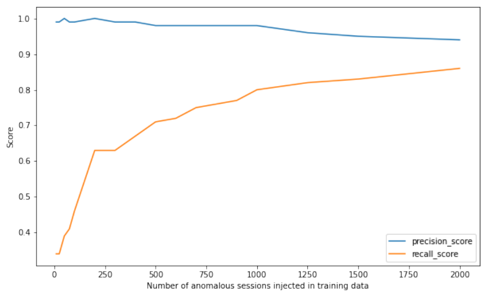
Figure 37: Precision and recall graphs for repetition pattern with deviation rate = -0.55, 0, 0.2 and 1



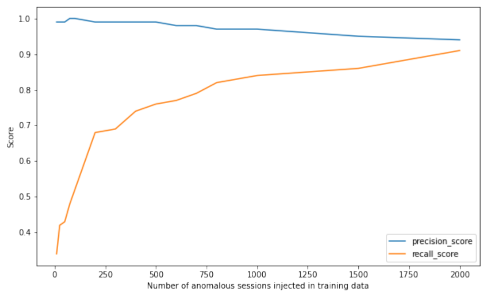
(a) Precision and recall scores for repetition pattern with deviation rate = 3



(b) Precision and recall scores for repetition pattern with deviation rate = 5



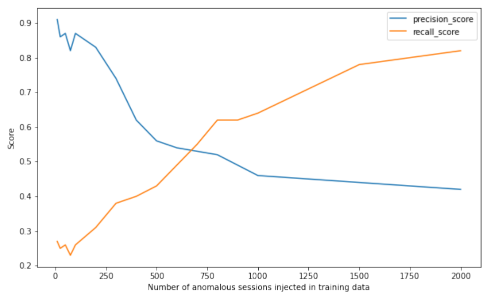
(c) Precision and recall scores for repetition pattern with deviation rate = 10



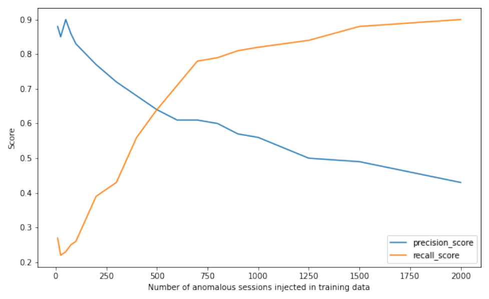
(d) Precision and recall scores for repetition pattern with deviation rate = 25

Figure 38: Precision and recall graphs for repetition pattern with deviation rate = 3, 5, 10 and 25

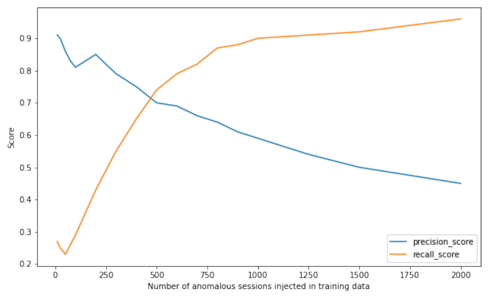
E.2 Precision recall plots for direct-follow pattern



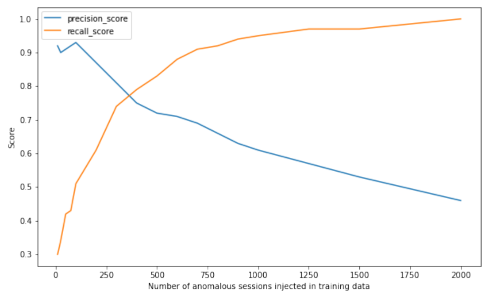
(a) Precision and recall scores for direct-follow pattern with deviation rate = -0.1



(b) Precision and recall scores for direct-follow pattern with deviation rate = 0

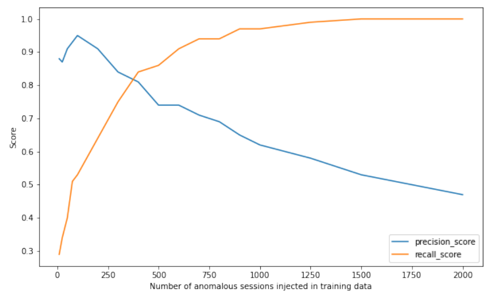


(c) Precision and recall scores for direct-follow pattern with deviation rate = 0.2

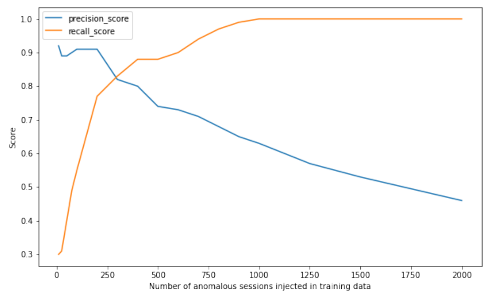


(d) Precision and recall scores for direct-follow pattern with deviation rate = 1

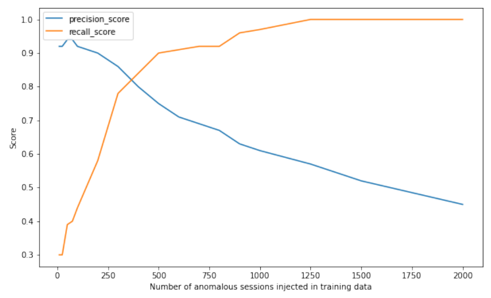
Figure 39: Precision and recall graphs for direct-follow pattern with deviation rate = -0.1, 0, 0.2 and 1



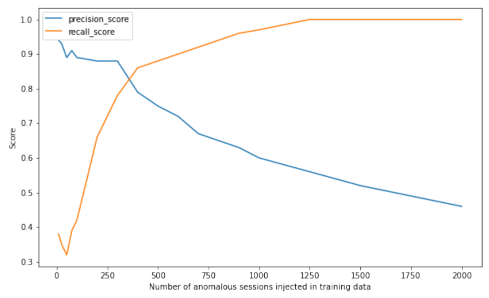
(a) Precision and recall scores for direct-follow pattern with deviation rate = 3



(b) Precision and recall scores for direct-follow pattern with deviation rate = 5



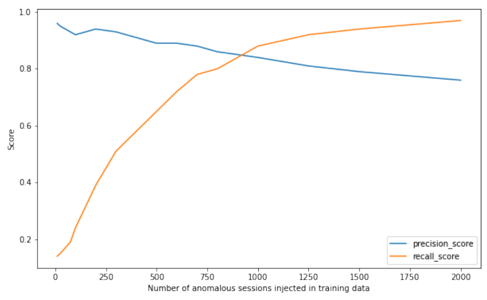
(c) Precision and recall scores for direct-follow pattern with deviation rate = 10



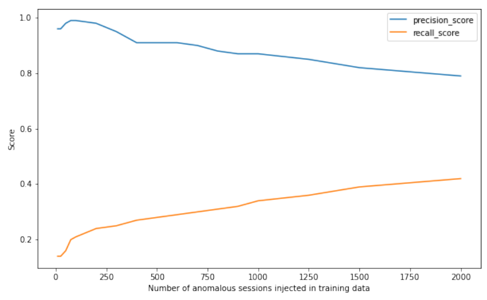
(d) Precision and recall scores for direct-follow pattern with deviation rate = 25

Figure 40: Precision and recall graphs for direct-follow pattern with deviation rate = 3, 5, 10 and 25

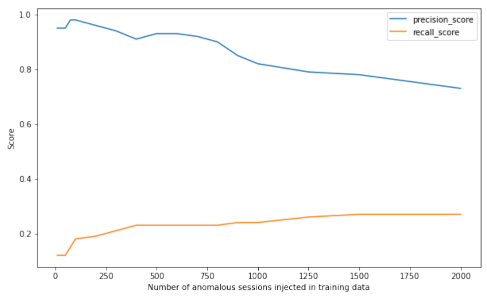
E.3 Precision recall plots for advanced repetition pattern



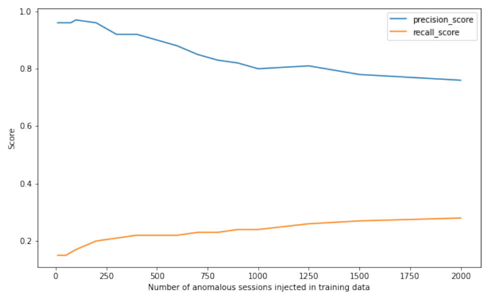
(a) Precision and recall scores for advanced repetition pattern with deviation rate = 0



(b) Precision and recall scores for advanced repetition pattern with deviation rate = 0.2

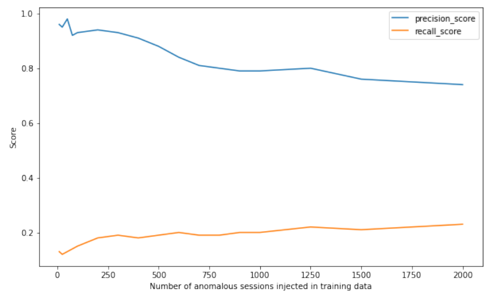


(c) Precision and recall scores for advanced repetition pattern with deviation rate = 1

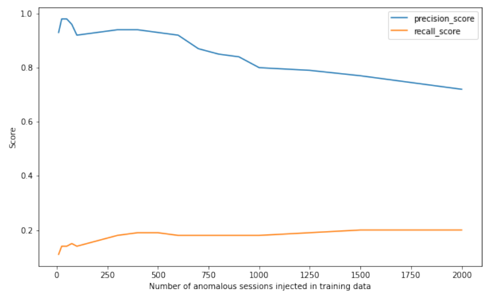


(d) Precision and recall scores for advanced repetition pattern with deviation rate = 3

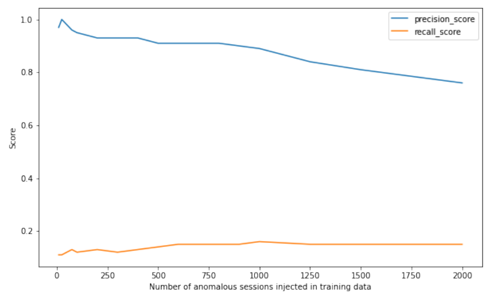
Figure 41: Precision and recall graphs for advanced repetition pattern with deviation rate = 0, 0.2, 1 and 3



(a) Precision and recall scores for advanced repetition pattern with deviation rate = 5



(b) Precision and recall scores for advanced repetition pattern with deviation rate = 10



(c) Precision and recall scores for advanced repetition pattern with deviation rate = 25

Figure 42: Precision and recall graphs for advanced repetition pattern with deviation rate = 5, 10 and 25