MASTER

Predicting the occurrence of an activity in the remainder of the customer journey

Wolters, L.J.

Award date:
2021

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science
Process Analytics

Predicting the occurrence of an activity in the remainder of the customer journey

Master Thesis

L.J. Wolters

Supervisors:
Dr.ing. Marwan Hassani
Marco Cordewener MSc

Eindhoven, October 2021

Abstract

Customer journey analysis is important for organizations to get to known as much as possible about the main behaviour of their customers. The customer journey is the sequence of interactions between a customer and an organization. The result of such customer journey analysis can be used to improve customer experience within their organisation. Customer journey analysis often uses customer journey maps which mainly focuses on comparing expected customer journeys with the main journeys executed by the customers. More complex investigations need more complex techniques. This thesis aims to fill one of the gaps by creating a framework that is able to predict the occurrence of a certain activity of interest in the remainder of the customer journey. The remainder of the customer journey are the interactions between the customer and the organizations that will take place in the future. By using process mining techniques the process describing the customer journeys is analyzed. The current process is a starting point in predicting the occurrence of a certain activity in a customer journey. This thesis provides a framework for high importance activity prediction at certain moments in the customer journey. This can be used to predict whether the current customer is expected to interact with a certain activity on a later moment in its journey. Different prediction models are researched to investigate which model is most suitable for the task of high importance activity prediction. This framework combines elements from customer journey analysis, process mining, machine learning and deep learning. The proposed pipeline is evaluated on a data set of VGZ, a Dutch health insurer, and the BPI2012 benchmark data set. The data set of VGZ contains customer journey data on a declaration process. The BPI 2012 challenge data log contains data of an application process for a personal loan or overdraft. The results highlight the usefulness of the proposed framework under realistic business settings and that it is general enough to be applied to various scenarios. The usefulness is shown as the predictions are improved over the baseline model. It is applicable to a realistic business setting as the framework is evaluated with customer journey data recorded during the execution of the process.

Keywords: customer journey analysis, process mining, process prediction, machine learning, deep learning

Acknowledgement

This master's thesis is the result for my graduation project for the master Data Science in Engineering. The project was carried out at the research group Process Analytics of the department of Mathematics and Computer Science of Eindhoven University of Technology (TU/e), Underlined and in collaboration with health insurer VGZ.

First of all, I want to thank my supervisors Marwan Hassani from the TU/e, Marco Cordewener from Underlined and Teun Sutmuller from VGZ for their guidance during my thesis. Furthermore, I want to thank everyone who supported me during my time as student. Special thanks to my father, mother, sister and brother who always believed in me and supported me when needed.

Contents

Chapter 1

Introduction

This chapter introduces the context of this thesis, describes the research problem and defines the research questions answered in this thesis. Section 1.1 describes the outline of this thesis.

Nowadays, customer journey analysis is useful for companies trying to understand how the customer interacts with the company. Next to understanding the customer journey it can also be used to improve the customer experience [6]. Customers can interact with a company over multiple channels, such as website visits and phone calls. Not all interactions provide the same customer experience and influence the customer satisfaction in the same way [41]. One of the options to investigate in customer journeys is by using customer journey maps (CJM) [6]. CJM are graphs that show the customer journey over time by indicating the touchpoints the customer has interacted with to achieve a certain goal. Such a goal can be to purchase a product or gather information about a certain topic. CJM can be used to gain insights in the main behaviour of customers interacting with the company. These insights can be compared with the expected journey. However, more complex questions to improve the customer journey require more complex techniques. Next to understanding the current customer journeys, it is also interesting for companies to predict whether these customers will interact with a certain touchpoint on a later moment in their journey. Knowing in advance which customer will encounter certain touchpoints, might provide the option to prevent interactions with touchpoints that are often experienced bad or might save resources. Such insights cannot be gathered with a CJM but require more advance techniques. Examples of such techniques are the application of process mining [1], machine learning [23, 7] and deep learning techniques [16, 33, 15, 12]. Current research has already shown interest in next event prediction and final outcome prediction for running customer cases [36, 34, 35, 17]. These methods all focus on predicting the first next touchpoint in a customer journey or the final outcome. A final outcome can for instance be whether a customer will purchase a product. Predicting the final outcome or next event is different from the research in this thesis. In this thesis, the research conducted will investigate whether the customer will interact with a certain activity in the remainder of its journey. Therefore, it should not necessarily be the next activity, even though it might in some cases be the next activity. Neither is it related to the final outcome. For example if the final outcome prediction is concerned with predicting whether a certain product will be purchased in a shop, the prediction of an interesting activity can be related to whether the customer wants more information about a product. This thesis provides the first effort to fill the gap in future touchpoint prediction that is neither the next activity nor final outcome. This is achieved by providing a repeatable framework for future high importance activity prediction. This thesis has been performed in collaboration with the Dutch health insurer VGZ. VGZ wants to retrieve insights in which customers are most likely to call VGZ. Performing a call is often experienced bad; therefore, it is interesting to prevent such interactions. A first step to prevention is knowing which customer will call. For this purpose, a data set containing declaration data of the customer is made available. The goal is to predict at a certain moment in the customer journey which customers will call VGZ in the remainder of their journey.

The solution proposed in this thesis uses process mining techniques to analyze the current customer journeys. The insights gathered serve as basis to indicate the decision moment and potential activity. For customer journeys reached the moment defined as the 'decision moment' it should be predicted whether the current customer will interact with the 'potential activity' in the remainder of its journey. Using the customer journeys, decision moment and potential activity, machine learning and deep learning models are trained to perform predictions. The solution provides a repeatable framework to predict the occurrence of a potential activity in a customer journey. The performance of the different prediction models will be evaluated in different settings. Next, the effect of defining the training, the validation and the test set to train and test the models is investigated. For this purpose, the sets are created either time based or randomly to investigate the difference in the results. A prediction model should stay up to date to recent customer journeys to be most useful. Therefore, research is conducted in the resources needed to keep a model up to date with respect to the quality gain. To investigate this, a sliding window and landmark window are used over the available data. This thesis addresses the next main research question:

How can future touchpoint occurrence of a certain activity of interest be predicted in a running customer journey based on process mining techniques?

The next subquestions are defined:

1. How can process mining techniques be exploited in the context of customer journey analysis?

2. What is the influence of the prediction model on the quality of the future touchpoint predictions?

3. How is the result of the prediction affected by the creation of the training, the validation and the test data?

4. How are the model training and prediction quality affected by applying various windowing techniques?

Answering the research question, the aim of this thesis is twofold:

1. Bridging the gap between process mining and customer journey analysis and using process mining techniques to improve the customer journey analysis.
2. Defining a repeatable framework for future touchpoint prediction in a customer journey.

1.1 Outline

This thesis is structured as follows: Chapter 2 provides an overview of related work in the context of this thesis. Chapter 3 contains notations required to understand the solution and explains the research problem in more details. Chapter 4 describes the proposed framework for high importance activity prediction in the customer journey and discusses the components of the framework in detail. Chapter 5 introduces two data sets, a performance measure to compare different models, a baseline model and evaluates the application of the high importance activity prediction framework on the two data sets. This evaluation entails the comparison of the different prediction models, the comparison of the time based or randomly generated training, validation and test set and the comparison of the windowing techniques. Finally, Chapter 6 concludes the research by summarizing the contributions of this research and proposes future research directions.

Chapter 2

Related work

This chapter provides pre-existing work that is related to this thesis. First, this chapter discusses related work from the fields of customer journey analysis in Section 2.1, followed by the field of process mining in Section 2.2. Section 2.3 discusses machine learning techniques and deep learning architectures that can be useful in the field of process prediction. This chapter concludes with Section 2.4 where related papers to the research problem are discussed.

2.1 Customer journey

This section introduces the concept of customer journeys, providing the definition and characteristics. The customer journey is the sequence of interactions between a customer and an organisation [25]. The customer journey contains the complete cycle of not yet being a customer to being a customer and potentially remaining a customer for forever. All intermediate interactions between a customer and an organisation define the customer journey. The first time a customer uses a service or purchases a product is the start of the customer journey. The journey continues as the customer uses/purchases a service or product [25].

The customer journey can be visualized in a customer journey map, in which each interaction of a customer with an organisation is defined by a touchpoint [18]. Challenges faced when visualizing all customer journeys are the large number of possible touchpoints and the different order over which the customers can encounter the touchpoints during its journey [6]. Figure 2.1 shows an example of a customer journey map in which the sequence of activities over time is shown on the horizontal-axis with the associated touchpoints on the vertical-axis. A single touchpoint represent a single interaction between the customer and the organisation. The complete path of one customer over time with multiple touchpoints is visualized with a line. The complete map shows all possible journeys taken by at least one customer. These maps are a systematic approach for organisations to understand the paths taken by customers over the different touchpoints. The visualizations may also provide insights into the customer experience at each touchpoint and how customers would like the customer experience to be [25]. The observations may be used to improve

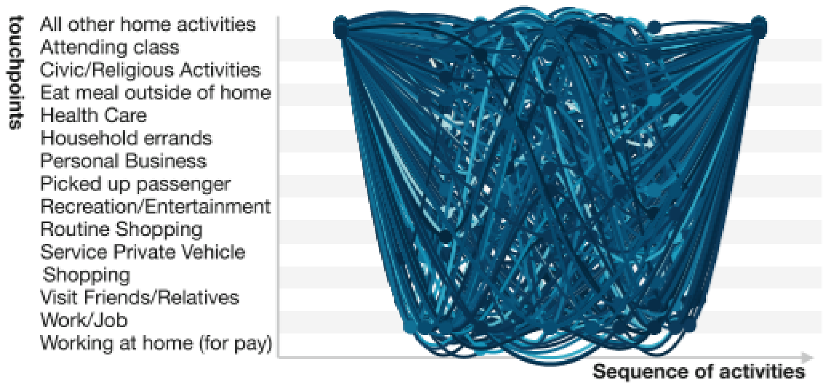


Figure 2.1: An example of a CJM, source: [6]

the current process such that the experience of customers is enhanced.

Even though customer journey maps can offer an overall view of the customer journey, they have to be created manually requiring a lot of domain knowledge and time [18]. Therefore, this thesis uses process mining to analyze the customer journey. Process mining allows modeling of the customer journey with an automated and data driven method, which speeds up the creation of the models.

2.2 Process mining

Process mining is a research discipline that falls somewhere between machine learning and data mining on the one hand and process modeling and analysis on the other hand [1]. Process mining seeks the confrontation between event data (i.e., observed behavior) and process models (hand-made or discovered automatically). The idea of process mining is to discover, monitor and improve real processes by extracting knowledge from event logs available in information systems. Process mining allows process owners to gain knowledge and insights in (business) processes by analyzing the event data stored during execution of the process by the users of the system [1]. Figure 2.2 shows how process mining establishes links between real data of actual processes and process models. Additionally, it highlights in red the three sub-fields of process mining [1] which are described below. Many of the available process mining techniques can be classified into one of these sub-fields; therefore, it provides a good characterization of the aspect that process mining aims to analyze. However, process mining is not limited to these subfields.

- **Process discovery:** A process model is produced from an event log without the use of any other information about the process. The model produced shall explain the behavior recorded in the log. Examples of models are petri nets or causal nets. In the case that the event log contains additional information about resources, one can also discover resource-related models. An example of a resource-related model is a social network, which provides information on how people interact within an

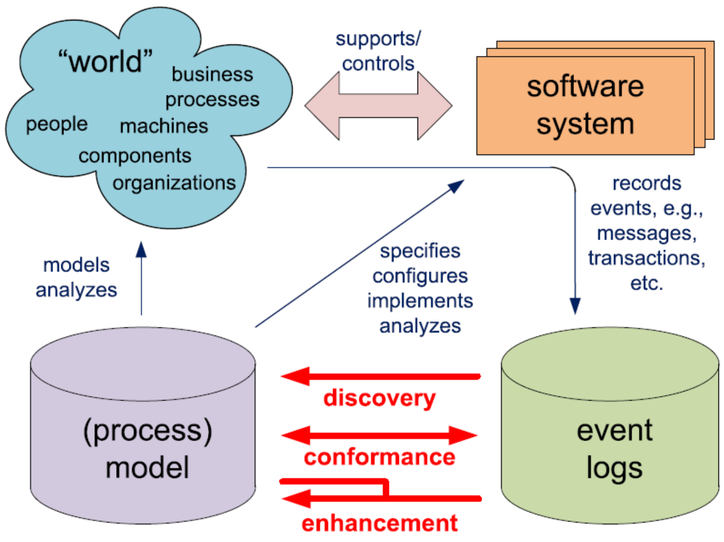


Figure 2.2: Positioning the sub-fields of process mining, source: [1]

organization.

- **Conformance checking:** For the purpose of conformance checking both an event log and a process model of the same process are required. This approach can be used to check whether the process captured in the model conforms to the behaviour recorded in the event log and vice versa. Hence, conformance checking may be used to detect, locate and explain deviations, and to measure the severity of these deviations.

- **Process enhancement:** The idea of process enhancement aims at changing or extending an existing model. To achieve this it needs a model and an event log from which it uses information about the actual process recorded in the event log. Two examples of enhancement are repair (modifying to model to better reflect the process captured in the log) and extension (adding a new perspective to the process model).

Event logs

The underlying basis for the algorithms of these three perspectives is an event log in which the real data of the actual processes is recorded. Therefore, proper events logs are required to apply process mining. The original data may be stored in different data sources, such as an Excel spreadsheet or a database table. However, the data may be stored over several data sources and does not have to be well-structured. Consequently, collecting relevant data may require a great deal of effort [1].

An event log consists of a set of events related to a single process. Each event is related to a case, with each case associated to an unique identifier. Multiple events may be related to the same case. The sequence of events in a case is defined as a trace. Furthermore, each event is related to an activity, referring to what has happened. A timestamp may be assigned to the event, indicating at which moment in time an activity has taken place. This timestamp may be used to order the sequence in the trace. Extra properties may be associated to an event, these are referred to as attributes. Examples of attributes are contact channel, status of a request or age.

CJM	**XES**
CJM	log
customer journey	case
interaction	event
touchpoint	activity
interaction timestamp	event timestamp
interaction attribute	event attribute

Table 2.1: Mapping from CJM to XES

The data considered in this thesis is customer journey data; therefore, it is important that customer journey data can be converted to an event log. Typically, event logs are stored in XES format. Table 2.1 shows that the components of a CJM can be mapped to a XES format. The mapping from CJM to XES enable process mining techniques to be applied to customer journey data.

Process prediction

Predictive business process monitoring is a new evolving technique in process mining [26]. The field of process prediction is less about providing process models, but focuses on single process instances. For an uncompleted process instance it tries to provide insight in the remainder of a process at run time. This prediction may be used to predict the next event of the trace [19, 31, 34], the expected outcome [17] or a suffix [35]. Section 2.4 provides more details on existing solutions for process prediction. These predictions may be used to identify unwanted behaviour during execution, preferably as earlier as possible. Expected unwanted process execution may be signaled and possibly the expected output may be influenced and improved already during run time.

2.3 Machine learning and deep learning models

This section provides background knowledge on machine learning and on deep learning methods interesting for the scope of the research in this thesis.

2.3.1 Machine learning

Machine learning provides automated methods for data analysis [23]. This set of automated methods can detect patterns in data and use these patterns to perform decision making under uncertainty. Given past data machine learning methods aim to predict future characteristics. Two main types of machine learning are considered. Predictive or supervised learning and descriptive or unsupervised learning. Supervised learning makes use of a training set, which contains input and matching outputs. The goal is to learn a mapping from the input to the output. This mapping can be used to predict the output for new inputs. Supervised learning is used for classification and regression models. In the case of classification models we have m classes to which an input can belong. The goals is it to predict to which of the m classes the current input belongs. In the case of regression models the variable that needs to be predicted is a continuous variable. The second type, unsupervised learning only uses a set of inputs. The goal of unsupervised learning is to find patterns in this input [23].

Next we will consider a machine learning algorithm for classification and regression.

Random Forest

A Random Forest (RF) model can be used for classification and regression and is an ensemble predictor constructed of a number of decision trees [7]. A decision tree is a hierarchical model, consisting of internal decision nodes and leaf nodes which contain the predicted label of an instance. Decision trees uses a divide-and-conquer approach to decide the correct label of an input instance [24]. Given input features an instance will be mapped to an associated label. Starting at the root of the tree, each internal decision recursively checks for a condition to hold and depending on the outcome chooses a branch to take until a leaf node is reached. The label associated with the reached leaf node is the prediction for the current input. Figure 2.3 shows an example of a decision tree to predict the flower species with internal decision nodes visualized with circles and decision leaf nodes with rectangles. Given the petal length and width, the flower is classified as a Setosa, Versicolor or Virginica. At the root node a decision is made based on the petal width. If the width <= 0.6 cm the current instance is classified as Virginica. If the petal has a width > 0.6 cm, the next internal decision node is reached. At this new node, the petal length is used for classifying the input as Versicolor if the length <= 4.9 cm and as Setosa otherwise. Each predictor decision tree in the ensemble model is trained with a sample of the training data. The RF model predicts the label of an instance based on its underlying decision trees. The label that has a majority vote over all decision trees in the ensemble is selected to be the final prediction of the RF. Using the majority vote improves the prediction performance of the RF model [7].

2.3.2 Deep learning

Deep learning (DL) is a sub-research field of machine learning that focuses on making accurate data driven-decisions by creating neural network (NN) models [16]. A NN is often organized into layers, where layers are interconnected and consist of a large number of computing elements called neurons [33]. Each input to a neuron has an associated weight. The input of a neuron are the output of the previous neurons together with the weights. An activation function is applied on the input to provide the output of the current neuron.

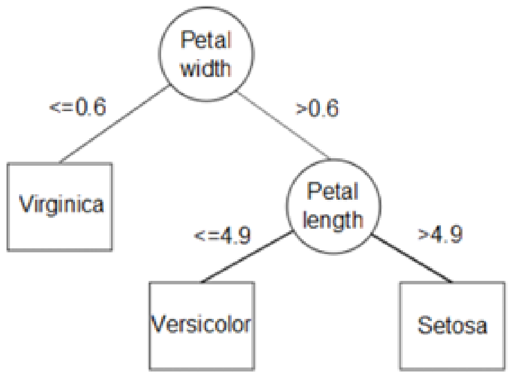


Figure 2.3: Example of a decision tree to predict the specie of flower, source: [9]

The initial input to the NN is the data. The neurons in the network process this input and modify this input to a desired output. The different weights increases or decreases the importance of an input variable to a current network cell; therefore, it also affects the outcome of the network. The optimal values of the weights should be determined by training the net on training data. A method to achieve optimal parameter setting is back propagation [39]. Figure 2.4 shows an example of an NN with three input variables and two outputs. It has one fully connected hidden layer and weights associated with its arcs.

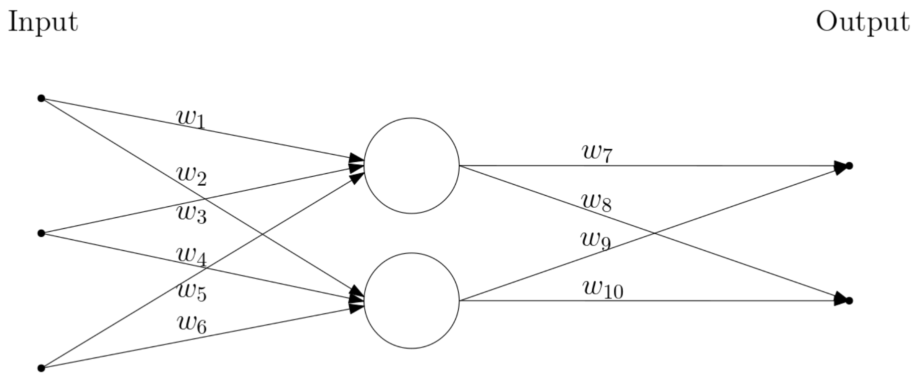


Figure 2.4: A NN with three inputs and two outputs

Next we will consider multiple types of deep learning methods.

Recurrent Neural Networks

Recurrent Neural Networks (RNN) are networks that can process sequential data of variable length [16]. The data points of the sequence are processed one at a time and the model may produce output at each step and has a memory state at each step. The RNN model combines information of multiple data points of different times to compute an output. For this purpose, the RNN model uses an internal hidden state. The result of the hidden state from the previous data point is used in the computation of the current data point. Figure

2.5 shows on the left side an overview of a RNN, on the right side the unrolled version of a RNN is shown. The square in the left figure shows the connection from the previous data point to the current data point. It is used to determine the value of the current data point. In the unrolled version, this is shown with arrows pointing from left to right to the *h* node one time step later in time. Just as with NN the model has parameters, such as weights on the arcs which require training to determine optimal values. Training a RNN is called backpropagation through time. Some of the parameters are shared as a RNN reuses the hidden state over the complete input sequence. Therefore, during the backpropagation the gradients are updated proportional over all future time steps. This may results in the problem of vanishing or exploding gradients. Therefore, it may be hard for a RNN to learn long-range dependencies in sequential data.

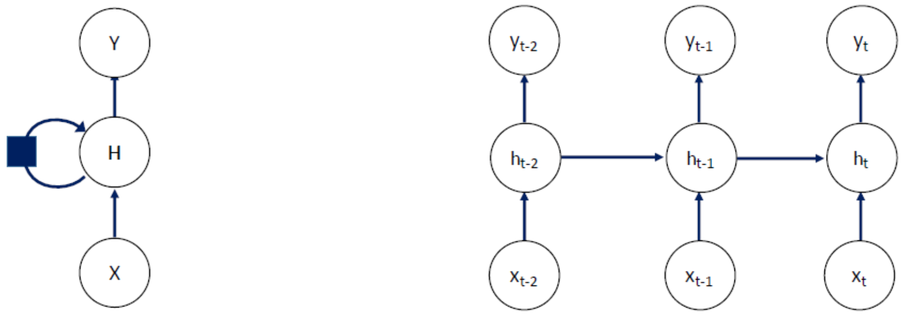


Figure 2.5: A representation of a RNN

Long Short-Term Memory

To overcome the former mentioned problem of the vanishing gradient a RNN is suffering from, one can use a Long Short-Term Memory (LSTM) network [15]. A LSTM network is a RNN architecture which is constructed to overcome the problem of not being able to learn long-term dependencies. For this purpose, a gradient-based algorithm is developed which enforces constant error flow through internal states. Constant errors mean that gradients will not explode nor vanish over time. To prevent gradients to vanish or explode gates are implemented in the hidden cells of the network. These gates are mechanism which allow a model to decide which information is kept and which information is forgotten. A LSTM cell contains three of these gates to control and protect the state of the cell, these gates are the forget gate, input gate and output gate. The forget gates is used to decide which information should be kept and what should be forgotten of the previous step. This decision is based on the previous cell state and the current input. The input gate decides which part of the new information should be remembered for a later state. The output gate determines the output which is based on the cell state and the input feature. It is used to decide when to access the information of the current cell to prevent other units from being perturbed by the output of another cell. Figure 2.6 shows an example of a LSTM cell with the gates. The left part in the cell represents the forget gate. The input gate is represented in the middle and the right part represents the output gate.

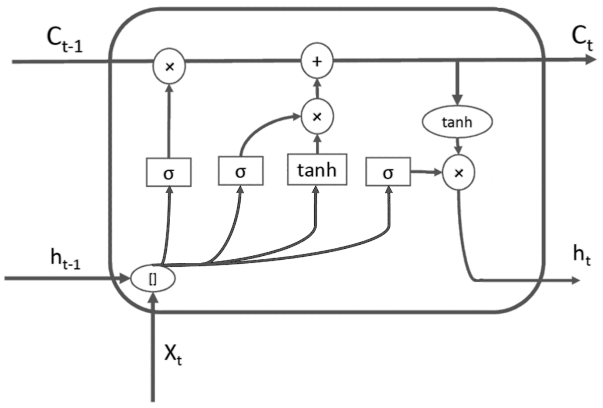


Figure 2.6: Representation of a LSTM cell

Generative Adversarial Network

A Generative Adversarial Network (GAN) is a generative model that trains via an adversarial process [12]. The network contains two models, the first one being a generative model and the second one being a discriminative model. An example of a GAN applied in the context of a trace suffix prediction is shown in Figure 2.7. In the figure the input is a trace. A trace is a chronological sequence of events, each event is denoted by e_x. The prefix of the trace are the first events of the trace up to event y. The suffix of the trace are the events occurring later than the prefix. The generative model captures the data distribution and generates new data that has the same data distribution as the original data. The discriminative model predicts whether a sample is originated from the original data or generated by the generator. Together, the generator and discriminator play a 2-player game in which the generator tries to fool the discriminator. In this 2-player game, the discriminator tries to maximize assigning the correct label to each data sample while the generator at the same time tries to minimize this same objective. Training a GAN can be done with backpropagation. The generator has accomplished its goal if the discriminator is unable to distinguish real instances from instances generated by the generator.

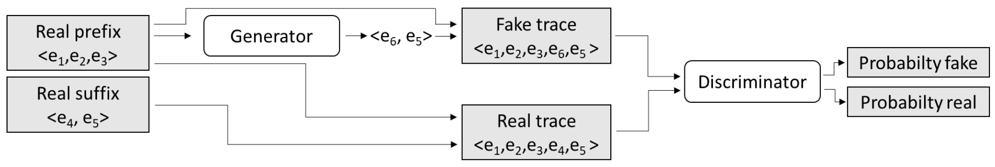


Figure 2.7: Representation of a GAN structure when applied in the context of a trace suffix prediction

2.4 Existing solutions for predictive analysis in process mining

Predicting next events and timestamps in a running trace have been studied before. Several works are worth to mention here because they are using interesting techniques or have similar goals. Though none of these works have exactly the same assumptions, data or goal as this thesis. This section describes several studies that are related to next event, next time stamp and remaining trace prediction.

The work in [31] and [19] both propose a way to predict a next event within an event sequence. The event sequence is the set of previous actions performed by a customer. Both papers use data of a online grocery shop in which the events are defined by customers adding products to their shopping basket. For the prediction task the set of products currently in the basket are considered, but not the order in which the products are added. Furthermore, the previous orders of the current customer are considered in order to determine which product is most likely the next product to be added. For the prediction task Rudin et al. [31] use association rule mining where rules are created based on a minimum confidence and support threshold. Letham et al. [19] access the prediction task as a supervised ranking problem, where events are ranked according to their likelihood of being the subsequent event in the sequence. The accuracy of their model is determined by the position in the list of the real next product. If the product is higher, the accuracy will also be higher.

These two techniques are not applicable for the research problem of this thesis as the context of the study is different. In these studies the previous interactions of the current customer on the current visit are considered as well as the interactions of the same customer on previous visits. In the context of the current thesis, previous interaction sequences of all other customers are deemed important. Furthermore, the order of the past event sequence is not considered. The events are considered as a set in which the order is not important. For the current research the order as well as the time between these activities will be considered in the prediction of the next event.

The work by Hassani and Habets [14] focuses on improving the process of current customers. They analyze whether it is possible to take proactive steps based on the expected next touchpoint of a customer. The first step to achieve this goal is to use process mining techniques to analyze the customer journey. The combination of historical available customer journey data and the insights of the process analysis is used to retrieve predictions on the next touchpoints. For this purpose of next touchpoint prediction, logistic regression, random forests, XGboost and LSTM classifiers are compared. In their research they showed that the random forest underperformed compared to the other three models. A possible reason for this underperforming provided in the research is that they did not tune the parameters of the random forest. In the research XGBoost was the best performing classifier for next touchpoint prediction in the customer journey. XGBoost also beat the random predictor baseline classifier.

The approach of Hassani and Habets [14] is not applicable in the current research. In this research the goal is to predict for a customer whether he / she will interact with a certain touchpoint in the remainder of its journey. This touchpoint does not need to be the next touchpoint. Therefore, only gathering information on the next touchpoint does not answer the research question. The idea of first retrieving an overview of the executed customer journeys by applying process mining techniques is interesting. Furthermore, if applying a RFC, the parameters should be tuned, just as with the other classifiers to train the best model possible.

Terragni and Hassani have proposed a technique to analyze and optimize the customer journey by applying process mining and sequence-aware recommendations [37, 38]. The proposed approach uses process mining to discover the process that describes the customer journeys stored in the customer log. They explain how the customer journey log containing customer journey data of website visits on a certain website can be converted to an event log. The event log is used in combination with process mining techniques to discover the process model explaining the behaviour captured in the log. The discovered process is used to find insight by comparing the current process and the expected process. Furthermore, the process of different clusters of customers is compared. Combining the results of this analysis the goal is to optimize key performance indicators and to improve the customer journey by providing personalized recommendations. The personalized recommendations are based on general user behaviour and earlier executed actions / visited website pages of the current customer that are used as implicit user feedback.

The goal of Terragni and Hassani is to predict what a customer will like. Examples are recommendations on products that a customer is likely to be interested in or movies that he / she may like. The goal in the current research is to predict what a customer will do. More specifically, the goal is to predict whether a customer will encounter an action that is often experienced badly. Predicting positive recommendations is different from predicting what actions a customer will perform. Therefore, the second part on personalized recommendations cannot be used for the current research. The idea on how customer journey logs are filtered and transformed to process models can be interesting. The research explains how the customer log is transformed to an event log and how process mining techniques are used to investigate the current process. Ideas on how to use process mining techniques in the context of customer journey data might be interesting to consider in the current research.

Tax et al. [34] proposes a technique based on a RNN with the LSTM architecture for next event prediction and its associated timestamp. Furthermore it predicts the event suffix of a running case and remaining time till the end of the trace. The suffix of the trace is determined by iterative applying the next event prediction, assuming that the prediction up to the current event in the suffix is correct. Events are encoded to categorical variables with one-hot vector encoding. The one-hot feature vector is combined with time-based features to more accurately predict the timestamp. The time-based features considered are based on the time since the previous event, the time since midnight and the time since midnight at the start of the week with respect to the current event. Furthermore, the paper proposes several architectures of the LSTM network, in which layers of the LSTM network can perform a single-task or may be shared for prediction. Figure 2.8 shows the three architectures researched. The left network has single-task layers, the middle network has shared multi-task layers and the right architecture shows a network with multi-task and single-task layers. The best results are gained in a model in which one layer is a shared multi-task layer and one single-task layer for both tasks separate containing 100 neurons per layer.

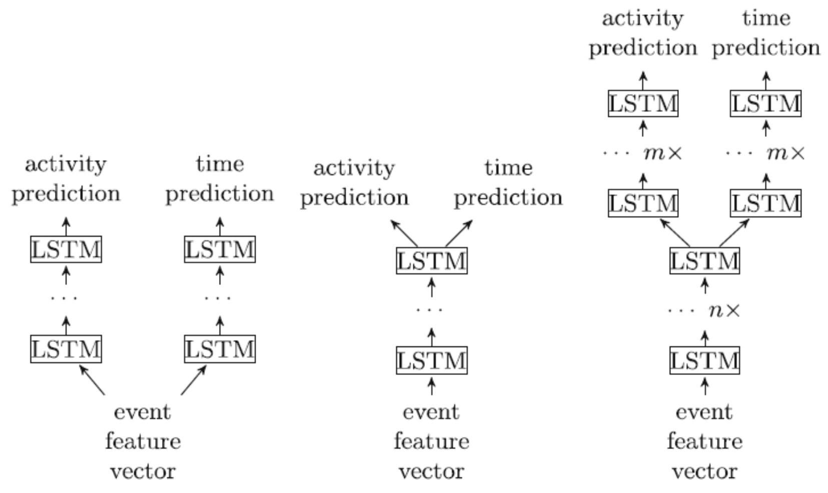


Figure 2.8: NN architectures research by Tax et al., source: [34]

This approach of Tax et al. encounters problems with traces in which the same event occurs multiple times as in that case the model predicts overly long sequences of that event. In the case of a health insurer, some events are expected to reoccur. Therefore, a solution for this limitation should be implemented to be applicable in the current context. Another issue that this paper suffers from is that the suffix is generated by iteratively predicting the next event. However, this approach suffers from propagating an error in a previous step to the next step. Which may result in a poor suffix quality. Due to these limitations the solution proposed by Tax et al. is not suitable for the current research.

The work by Taymouri et al. [36] uses a GAN architecture for the prediction of the next activity and next timestamp in a trace. Figure 2.9 shows the architecture proposed for next event prediction. In this work, the next timestamp is defined as predicting the time that will elapse between the current event and the previous event. The network considered consists of a discriminator and a generator, which both try to maximize its own outcome and minimize the outcome of the opponent. Both the discriminator and the generator consist of a LSTM network and feedforward layers. The discriminator has an extra feedforward layer to assign probabilities on whether the input is a ground-truth or a generated sequence. At convergence the generator is able to confuse the discriminator. The network trained using a GAN with LSTM networks will result in a prediction that is at least a good as predicting only with a RNN with a LSTM architecture. To deal with categorical activity variables, these are encoded with an one-hot vector encoding. Furthermore, Taymouri et al. provide predictions on several prefix lengths. The result shows that a longer prefix results in a prediction with a higher accuracy.

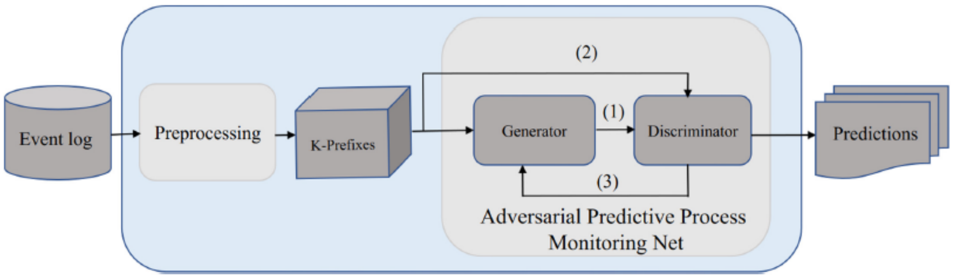


Figure 2.9: NN architectures of Taymour et al. for next event prediction, source: [36]

Only predicting the next event in a trace is not providing enough information for the research problem of this thesis. The goal of the current research is to predict whether an activity will occur in the remainder of the trace predicted on a certain moment in the trace. However, it should still be predicted that the activity occurs if other activities take place in between the current activity and the activity which should be predicted. Therefore, information only on the next activity does not provide enough detail to answer the research question.

In a later work by Taymouri et al. [35] an encoder-decoder GAN is addressed for suffix generation and remaining time prediction. The architecture for predicting the suffix and remaining case time is shown in Figure 2.10. In this case the generator consists of the encoder-decoder structure where both the encoder and decoder are represented by a LSTM network and the discriminator is also represented by a LSTM network. The encoder-decoder structure of the generator allows the creation of variable length suffixes. Based on the prefix, the generator is trained to model the complete suffix directly instead of only predicting the next event. Complete ground truth suffixes are used during training to learn the relationships and orders between activities of the prefixes and suffixes simultaneously. In the GAN structure the generator provides the suffix and remaining time of a prefix and the discriminator determines whether it is a realistic suffix and remaining time. As soon as the discriminator is unable to discriminate real and fake instances, the generator is able to create accurate suffixes. Next to predicting the most likely suffix, the paper proposes beam search to generate the k most probable suffixes for a certain prefix.

The technique for suffix prediction of Taymouri et al. is not directly applicable for the research problem of this thesis. Taymouri et al. create multiple prefixes and suffixes of one trace. Meaning that they do not consider a specific moment in the trace to predict the suffix. In this thesis, the goal is to predict at a certain moment in the trace whether an activity will occur in the suffix. Additionally, the predicted suffixes are always up to the end of the instance, without considering certain events of greater importance. Examples of such events could be events that indicate that certain activities of higher interest have occurred with respect to other activities. The current research wants to predict whether a certain activity will occur; therefore, we only need to predict the suffix up to the event indicating that that activity has occurred. Only if the activity of interest does not occur

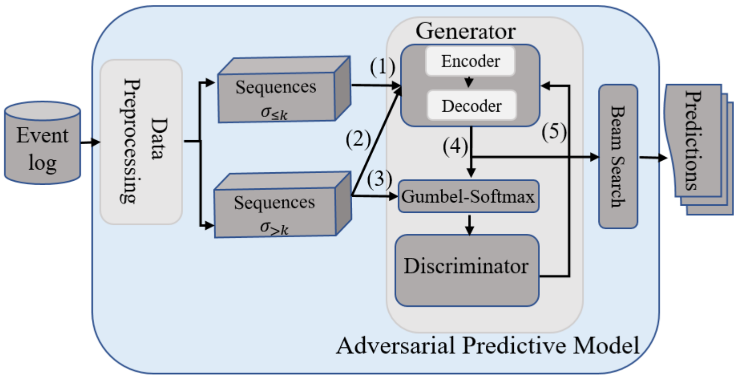


Figure 2.10: NN architectures of Taymour et al. for suffix prediction, source: [35]

in the suffix, we should predict the suffix up to the end of the trace. However, the idea on suffix prediction is very interesting for the goal of this thesis. Therefore, the approach can serve as a starting point for a solution to the research question of this paper.

Kratsch et al. [17] compare different machine learning and deep learning techniques on multiple event logs to determine which technique is most suited for outcome prediction. The machine learning techniques considered are support vector machines (SVM) and RF. The deep learning techniques considered are DNN and LSTM networks. Parameters of all four models are tuned with random search-based optimization combined with tenfold cross-validation. Original traces of the event logs are used to label the traces with the final outcome. Further preprocessing entailed preserving traces up to a certain event x. In the research values of 1 to 10 are all tested for x. For the traces the events as well as the attributes up to that event number in the trace are kept and used for the final outcome prediction. With the research Kratsch et al. showed that in general DL classifiers outperform ML classifiers on the accuracy and the F1-score of the predictions. The difference in performance decreased if the underlying data set was skewed. In the case of skewed data the ML classifiers had comparable results to the DL classifiers. Therefore, in the case of skewed data it might be beneficial to still use ML classification techniques. Because the gain in performance does not outperform the higher investment in DL techniques. As they have applied the research on different event log types the result may be considered to hold for event logs outside their research as well.

Predicting the final outcome of a case is not the same as predicting whether a certain activity will occur. The occurrence of such an activity does not need to be related to a final outcome. However, it should be possible to adapt the final outcome to high importance activity prediction. Additionally, the technique of Kratsch et al. define the decision moment as the moment that x events have occurred in the trace. In the current research, a prediction should be provided as soon as a certain proposition hold for that trace. This proposition might be valid after a different number of events have taken place in the trace; therefore, it is not possible to define a number of events after which the prediction should be provided. For this reason, the technique of final outcome prediction of Kratsch et al. is not applicable in the current research, but the technique of preprocessing the data and considering attributes might be interesting to consider.

Chapter 3

Problem exposition

This chapter describes the notations needed to understand the framework addressed in Chapter 4 and describes the research problem in more details. Section 3.1 defines the notation used in the remainder of the thesis and Section 3.2 explains deeply the problem addressed by this thesis.

3.1 Preliminaries

This section defines the notation used in the remainder of this thesis. Let

$$\mathcal{CJ} = (cj_1, cj_2, ..., cj_n) \tag{3.1}$$

be a log containing the customer interactions of all customers with a company. Each row in the log

$$cj_r = (cu_j, t, i, ia_1, ..ia_m) \tag{3.2}$$

defines a single interaction of one customer. The customer is identified with an unique customer identifier cu_j. The interaction was conducted at time t and the customer performed touchpoint i. The interaction of the customer may have interaction attributes $(ia_1, ...ia_m)$ defining extra information of the interaction. From $\mathcal{CJ}$ the set of all customers $CU = \{cu_1, cu_2, ...cu_{|CU|}\}$, the set of all types of touchpoints $I = \{i_1, i_2, ..., i_{|I|}\}$ and the set of all interaction attributes $IA = \{ia_1, ia_2, ...ia_{|IA|}\}$ can be extracted. The $\mathcal{CJ}$ will be converted in an event log, as explained in Section 2.2. Let

$$\mathcal{L} = (e_1, e_2, ...e_n) \tag{3.3}$$

be the event log of the customer journey. Each row in the log

$$e_r = (c_i, t, a, d_1, ..d_k) \tag{3.4}$$

defines a single event performed by one case identifier. The case is identified with an unique case identifier c_i and each customer $cu_j \in \mathcal{CJ}$ can be mapped to a c_i. The touchpoint of the interaction of the customer is renamed to an activity a and the activity is performed at time t. Each touchpoint i will be mapped to an activity a, but multiple touchpoints may map to the same activity a. Furthermore, events can have attributes $d_1, ...d_k$ defining extra information of the event and these attributes can be extracted from $ia_l \in IA$. From $\mathcal{L}$, the set of all activities $A = \{a_1, a_2, ..., a_{|A|}\}$, all case identifiers $CI = \{c_1, c_2, ...c_{|CI|}\}$ and the set of all attributes $D = \{d_1, d_2, ..., d_{|D|}\}$ can be extracted. The log $\mathcal{L}$ contains all traces of the customers in CI. Let

$$\sigma_i =< e_1, e_2, ..., e_{|\sigma_i|} > \tag{3.5}$$

define the trace of case identifier c_i. The trace contains the sequence of all events of case c_i ordered chronologically in time. For a trace the α-prefix is defined as the trace up to and including the first α events. The suffix is defined as event $(\alpha + 1)$ until the end of the trace. For the trace $\sigma_i =< e_1, e_2, e_3, e_4, e_5 >$ the 3-prefix $=< e_1, e_2, e_3 >$ and the 3-suffix $=< e_4, e_5 >$.

In the remainder of this thesis, the terms decision moment(s) and potential activity are used. The decision moment(s) are propositions which could be valid at a certain moment of the trace. It should be possible to check based on the events in the trace whether the proposition holds. Therefore, the proposition can exist of activities, elapsed time and recorded trace attributes. For example a proposition defining a decision moment could be the first occurrence of activity $x \in A$, as the activities are stored in the event. In the case that resources are stored as attributes, the proposition might be the first time that resource $y \in D$ was involved at the execution of event $x \in A$. The potential activity is an activity of the set A.

3.2 Problem formulation

This section explains the problem addressed in this thesis in more details. As already stated in the introduction, the aim of this thesis is two-fold:

1. Bridging the gap between process mining and customer journey analysis and using process mining techniques to improve the customer journey analysis.
2. Defining a repeatable framework for future touchpoint prediction in a customer journey.

The aim of analyzing the customer journey is to use the discovered insights to improve the customer experience. This can be achieved, for example, by adapting a process, such that it is more in line with the expectations of a customer. These adaptions can include the prevention of interacting with touchpoints that are classified as providing a negative experience or the prevention of long waiting times. Preventing customers to perform an activity which may provide a negative experience, provides a more satisfied feeling for the customer. This satisfied feeling, may be beneficial for a company, by the fact that a customer will stay at the current company and not switch to another company. To prevent customers having bad experiences, first it should be known which touchpoints contribute to these experiences. Furthermore, it should be investigated which customer based on the customer journey will most likely include this touchpoint in its journey. Additionally, it should be predicted as early as possible in a customer journey that a customer will encounter this touchpoint on a later moment in the journey. By knowing this as early as possible, a company can influence the customer journey, for example, by providing different information to prevent that the customer will get a negative experience. Next to improving customer satisfaction, preventing long waiting times may save resources. In the case that a certain action takes a lot of resources, and it is known early enough whether this action will be executed, it might be possible to schedule the resources in such a way that there will be less of a delay. This may again result in a smoother process and a more satisfied customer.

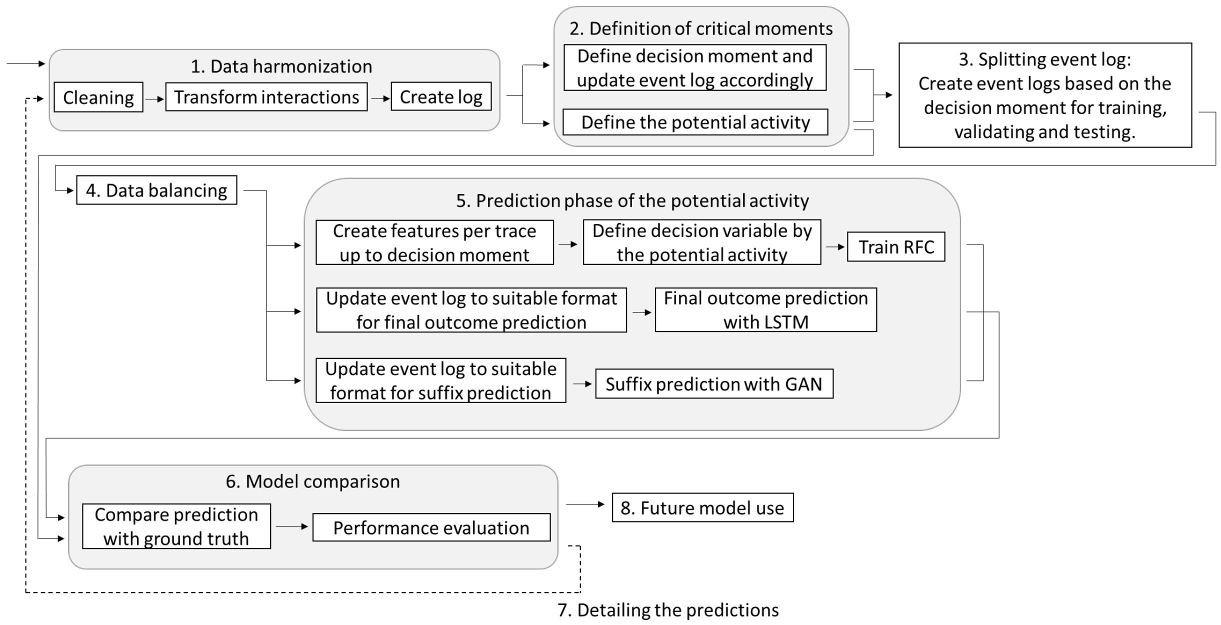
The first step in improving customer experience is to retrieve insights in the current customer journeys. To this end, process mining techniques will be applied on the event log extracted from the customer journey log. This will provide insights in which customers perform the potential activity and at what moment in their journey. The potential activity, is the activity of which the company is interested to know whether a customer will interact with this activity on a later moment in its journey. Next, a decision activities can be chosen at which it should be predicted whether a customer will in the future perform the potential activity. It would be beneficial for a company as such decision moment(s) are as early as possible in the trace. This part of the solution will contribute to the first contribution of bridging the gap between process mining and customer journey analysis to improve the analysis of customer journeys.

When the potential activity and decision moment(s) in the trace are defined, the goal is to predict for future cases whether the customer will interact with the potential activity as soon as the proposition defined as the decision moment is satisfied. Based on decision moment x, we know the x-prefix $< e_1, ...e_x >$ of a customer. The x-prefix contains information on activities executed by the customer and attributes associated with these activities. Based on the information in the x-prefix, the goal is to predict whether potential activity y will occur in the x-suffix of the customer. Where the x-suffix is $< e_{x+1}, ..., e_{|trace|} >$. As already stated in Section 2.4 there exists several techniques for next event [19, 31, 34], final outcome [17] and suffix prediction [35]. These techniques do not solve the research in this thesis directly. For this reason, this thesis will provide a framework to predict whether a customer will encounter a high importance activity in the remainder of its journey. The prediction can be used to retrieve insights in the customers who eventually will encounter the potential activity. This information might be used to improve the customer journey. By using machine and deep learning techniques predictions are made on whether a certain customer will indeed interact with the touchpoint that may be experience negative or cost a lot of resources. The repeatable framework for the second contribution for achieving the predictions will be described in Chapter 4.

Chapter 4

High importance activity prediction framework

This chapter introduces the high importance activity prediction framework (HIAP) to predict the occurrence of an interesting touchpoint in the remainder of the customer journey based on the journey up to a specific point in time. The prediction uses information of the event log prefixes and possible customer information to predict for a specific customer whether he/she will have a specific interaction in the future. Figure 4.1 shows a visual representation of the HIAP. The figure visualizes the steps that are conducted to retrieve interesting touchpoint predictions. Each step in the framework is numbered



1. Data harmonization
Cleaning
Transform interactions
Create log
2. Definition of critical moments
Define decision moment and update event log accordingly
Define the potential activity
3. Splitting event log: Create event logs based on the decision moment for training, validating and testing.
4. Data balancing
5. Prediction phase of the potential activity
Create features per trace up to decision moment
Define decision variable by the potential activity
Train RFC
Update event log to suitable format for final outcome prediction
Final outcome prediction with LSTM
Update event log to suitable format for suffix prediction
Suffix prediction with GAN
6. Model comparison
Compare prediction with ground truth
Performance evaluation
8. Future model use
7. Detailing the predictions

Figure 4.1: Methodology overview

by the subsection in which it is addressed in the current chapter. For example step '1. Data harmonization' is discussed in section 4.1. The first step (1) is concerned with data harmonization, consisting of three sub-steps: cleaning the customer data, transforming attributes and creating the event log and is described in Section 4.1. The next step (2) is related to defining the moment up to which the prefix will be available for prediction and to defining the potential activity. The potential activity is the interesting activity of which it should be predicted whether this will occur in the remainder of the trace. Defining the two critical moments can be executed in parallel and will be elaborated in Section 4.2. When both these steps are completed, the results will be combined to create separate event logs (3) used for training, validating and testing models to predict the occurrence of the potential activity in the traces. The creation of a training, a validation and a test log is described in Section 4.3. Section 4.4 discusses techniques (4) to balance the event log data. The next step (5) is concerned with finalizing the data for specific models and training models to predict whether the customer will encounter the interesting interaction in the future. This step is divided into a RFC, LSTM and GAN model, all three models are discussed in Section 4.5. Section 4.6 is related to (6) comparing the results of the prediction models and the ground truth and comparing the results of the different models. After completing an iteration of the framework, a new iteration (7) can be applied for more into-depth research, this process is described in Section 4.7. Last, Section 4.8 describes (8) how the preferred model can be used in the future to perform predictions for new customer journeys.

4.1 Data harmonization

Customer journey data may be stored in different sources. Furthermore, customer data and event log data almost always contain unnecessary data or missing data. This data may mislead further analysis and produce low quality results [1]. Therefore, data harmonization has to be conducted to combine the data of the different sources and adapt the data such that the quality of the data will improve. The exact data harmonization techniques may differ for each data set [2]. The goal of this step is to create a preprocessed event log that can be used for further research. This part of the framework contributes to answering the first research subquestion on how process mining techniques can be exploited in the context of customer journey data. It is the case that customer journey data first has to be transformed to an event log to be able to apply process mining techniques. This section will explain some options of data harmonization, but different scenarios may require different techniques to be applied.

The first step of data harmonization is cleaning the data set. Cleaning includes filtering incomplete journeys and filtering unnecessary attributes or touchpoints. For some customer journeys, it is possible to define touchpoints that define the end of a customer journey. For example, if the goal of a customer is to buy a product, the current journey can be defined as finished if the customer has received and paid the items he/she has purchased. For the customer journeys that do have items in their shopping basket, but do not have received the items yet, the journey can be defined as a running case. When considering the customer journey, the best results will be achieved by only considering completed cases. Therefore, only cases which end with a touchpoint that defines the end of the current journey should be considered. Furthermore, it might be the case that some of the recorded touchpoints and/or attributes are not of interest for the current research. These touchpoints and/or attributes can cause noise in the research of interest if these will be left in the customer log. Therefore, these touchpoints and/or attributes should also be filtered out.

The next step of data harmonization is the transformation of interactions. By transformation new interaction, timestamp and attribute combinations are derived. As information may be stored in different tables, it might be needed to combine different elements to retrieve complete information. For example, assume that one table stores the action with the timestamps that a customer has received a product. Another table stores the order in which products are sent to the customer but without timestamp. Combining this information can introduce one new interaction stating at which time which product is received by the customer. The transformation of interactions can provide more concrete interactions, which might be helpful in the remainder of the research conducted.

The last step of data harmonization is the transformation of a customer journey to an event log. Section 2.2 already explained a possible mapping from a customer journey log to an event log. Each customer journey should be mapped to a case identifier, each touchpoint to an activity, each interaction timestamp to an event timestamp and each interaction attribute to an event attribute. A mapping should exist for each element of the interaction to the event. Additionally, it is possible to define an element different from the customer identifier to serve as the new case identifier. Most commonly, the mapping from customer log to event log is by mapping each customer identifier to a case identifier. However, it is possible to choose another attribute, on which cases might be defined. For example, let the customer log contain online purchases of customers to a shop. Let each touchpoint be associated to a customer identifier and a purchase identifier. In this case, each trace can be identified by a customer, but it is also possible to define a trace per purchase of a customer. Both identifiers can be chosen as case identifier for the event log, but in all cases a mapping should be defined from identifier in the customer log to the case identifier in the event log. Furthermore, each touchpoint should be mapped to an activity. This might be a one-to-one mapping, where each activity represent one touchpoint, but it might also be a many-to-one mapping, where an activity might represent multiple touchpoints. A one-to-many mapping is impossible, as one touchpoint cannot be represented by multiple activities to prevent certain touchpoints to occur multiple times. The same applies for timestamp and attribute mappings.

4.2 Definition of critical moments

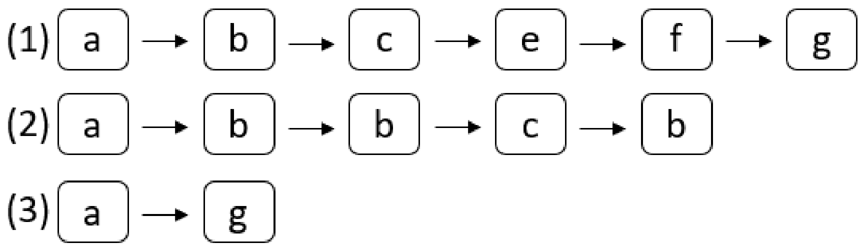
As soon as the event log $\mathcal{L}$ is extracted from the customer journey log $\mathcal{CJ}$, the process model captured in the $\mathcal{L}$ can be discovered. This model might be used to retrieve insights of the current customer journeys and to provide a basis for defining the critical moments. This step of the framework again contributes to the first subquestion on how process mining techniques can be exploited in the context of customer journey analysis. As at this step the process mining techniques are applied to the event log extracted from the customer journey data. The critical moments in the customer journey are the decision moment and the activity of interest called the potential activity. To discover the process model different process mining discovery algorithms can be used. Each discovery algorithm creates it own output model, examples of visualization are petri nets [4] and directly follow graphs [1]. The models should be able to capture the behaviour seen in the event log, at the same time, the models should also not generalize the process too much. If a model generalizes too much, the model allows for much more behaviour than the behaviour captured in the log. An example of a model that generalizes too much is a flower model [1]. In the case of overgeneralizing, the created model does not provide a good overview of the current process captured in the log. Furthermore, the model should also be simple enough to be comprehensible. Generalization and simplicity provides some trade offs during the creation of the process models. Next, the definition of the decision moment and potential activity are discussed.

4.2.1 Decision moment definition

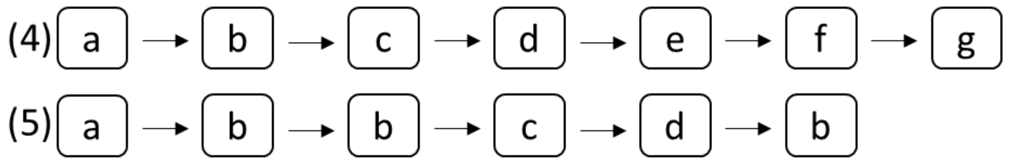
As already discussed in Chapter 3.2, the goal is to predict whether a certain activity will occur based on a predefined moment in the trace. This specific moment can be defined either by a specific activity or by a proposition based on the events in the trace that should hold. For example, the moment can be defined as the first time that activity x occurs or as soon as activity x and y have occurred. The first time that such an activity occurs or the proposition holds will be taken as the decision moment of the trace. As the prediction takes place at a certain moment, only the traces that at some moment satisfy the condition of a decision moment should be considered.

To determine the activity or proposition to define the decision moment, domain knowledge is of added value. Domain experts know the moment in a trace that is the most interesting point to provide the prediction. When determining the decision moment two criteria are considered. First, the goal of the prediction is to be able to adjust the remainder of the trace and prevent the occurrence of a certain activity or to be able to save resources. To be able to adjust the remainder of the trace or to save resources the prediction should be early in the process. The earlier it is known that a certain trace will eventually lead to an activity, the more time is provided to adapt the process. Second, the prediction should be as accurate as possible. Inaccurate predictions might cause adaptions to processes that do not need adaption or might cause that a case is not considered, while it will encounter the activity of interest. In general, more accurate predictions can be provided at the moment that more information is available about the current process. Therefore, a balance should be found between choosing an early decision moment and the accuracy of the prediction [20]. Both criteria on earliness and accuracy should be considered while determining the decision moment.

An example to show how the traces should be updated after the decision moment has been defined is provided in Figure 4.2. Figure 4.2a shows three traces of the original log. The traces are visualized by the activities belonging to the events of the traces. In



(a) Traces in the original log visualized by the activities of the events



(b) Traces in the log after inserting the decision moment visualized by the activities of the events

Figure 4.2: Log update to insert the decision moment in the traces visualized by the activities of the events

this example, assume that the decision moment is defined as the first time that activity *c* occurs. Two of the traces in the original log contain activity *c*, while the third trace does not. Therefore, the third trace should be removed from the log as it does not contain a moment on which the prediction should be performed. For traces one and two, the decision moment activity should be inserted in the event log directly after activity *c*. Let activity *d* be the activity representing the decision moment. The decision moment is added for easy reference to the moment at which the prediction should take place. This prevents rechecking at which moment the proposition defining the decision moment holds. This is useful for proposition that requests multiple requirements to be met. Figure 4.2b shows the updated traces after defining the decision moment in the traces.

The approach to insert the event defining the decision moment is explained by Algorithm 1. Next do updating the event log, a list is created in which the time of the decision moment and its associated trace identifier is stored for each trace in which proposition Z holds at some moment. The algorithm creates a subset of the event log, by iterating over all traces in the log. For each trace it checks whether the proposition for the decision moment holds at a certain moment in the trace. If the proposition is true for a trace in the log, it searches for the first event at which it holds and extracts the timestamp of that event. The step of checking at which moment in the trace the proposition is valid, is applied in Lines 4 and 5. Line 6 inserts the decision event in the log and Line 7 insert the trace identifier and the decision time in $DecT$. The decision event provides easy access to the trace attributes that are available during the prediction. As at the prediction only information of the trace up to the decision moment is available. The available information can easily be extracted by only using the trace information up to the decision moment. If

Algorithm 1: Updating event log with decision moment

```
Input: Event log $\mathcal{L}$, proposition $Z$ defining the decision moment
Output: Updated event log $\mathcal{L}$ and list $DecT$ containing the trace identifiers and
        decision times
1  $DecT = [\,]$
2  for $trace\ t \in \mathcal{L}$ do
3      if $Z$ holds somewhere in $t$ then
4          x = first event at which $Z$ holds
5          Time $time$ = x.time
6          $t.insert(t.traceIdentifier, time + 1sec, d)$
7          $DecT.insert(t.traceIdentifier, time + 1sec)$
8      else
9          remove $t$ from $\mathcal{L}$
10     end if
11 end for
12 return $\mathcal{L}, DecT$
```

the proposition is invalid at all moments in the trace, the trace is removed from the log at Line 8. Algorithm 1 ends when it has iterated over all traces and returns the updated event log $\mathcal{L}$ and list *DecT*.

4.2.2 Definition of the potential activity

Next to defining the decision moment, the potential activity (PoAc) has to be defined as well. Together, these can be used to decide in which traces the PoAc still occurs on a later moment than the decision moment. Determining the traces that contain the PoAc on a later moment than the decision moment is explained in Section 4.3. This PoAc is the activity of interest, about which it is preferred to know whether it will occur in the current customer journey. Domain knowledge is of added value in defining the PoAc. The occurrences of the PoAc will be predicted at the decision moment defined in Section 4.2.1. For example, let the traces in Figure 4.2b be event log $\mathcal{L}$ and let activity f be the PoAc. Let $|d|$-prefix be the prefix of the trace up to the event defined by activity d. Let $|d|$-suffix be the suffix of the trace with respect to the event defined by activity d respectively. In this case the suffix of trace four contains activity f, while the suffix of trace five does not.

As the occurrence of the PoAc will be predicted at the decision moment, the decision moment should be a proposition that is met earlier in the trace than that the PoAc occurs. However, it may be an activity that is occurring at a random moment in the suffix of the trace with respect to the decision moment. Therefore, it is not necessarily the next event or the last event of the trace. Since it could be unrelated to the final outcome of a process. For example, let a log $\mathcal{L}$ define the process of people buying a house. This process will entail researching the internet for available houses, planning a house visit, viewing the house, making an offer and buying the house. The final outcome of this process will be the purchase of a house, but a PoAc to predict might be whether an offer will be made on the house.

4.3 Splitting event log

Once the decision moment and the PoAc are defined, it is possible to create a set containing trace identifiers of the traces which contain the PoAc at a later moment than the decision moment and a set which contains the other trace identifiers. The method to create these two sets is explained by Algorithm 2. Lines 1 and 2 define two empty sets. Then the algorithm iterates over all traces in log $\mathcal{L}$. For each trace it determines the $|d|$-suffix, being the suffix of the trace with respect to the activity defining the decision moment, and checks whether the PoAc is an element of the suffix. In Lines 6 and 8, the trace identifier is added to the set based on the occurrence of the PoAc in the suffix of the trace. The set defining the traces in which the PoAc occurs in the $|d|$-suffix and the set defining the traces in which the PoAc does not occur in the $|d|$-suffix are returned by the algorithm.

Algorithm 2: Create sets of trace identifiers

```
Input: Event log $\mathcal{L}$, potential activity $p$
Output: Two lists $id_\alpha$, $id_\beta$ storing the identifiers where the suffix contains $p$ and
        the identifiers where the suffix does not contain $p$ respectively
1  $id_\alpha = \{ \}$
2  $id_\beta = \{ \}$
3  for $trace\ t \in \mathcal{L}$ do
4      $suffix_d = |d|$-suffix of trace $t$
5      if $p \in suffix_d$ then
6          $id_\alpha.append(identifier\ t)$
7      else
8          $id_\beta.append(identifier\ t)$
9      end if
10 end for
11 return $id_\alpha, id_\beta$
```

The next step is to create a training, a validation and a test set. These sets will be used for training, validating and testing prediction models later in the process. The training data set is the sample of the data that will be used to fit a model [30]. These samples are used to determine model parameters, such as weights in a neural network or decision variables of decision nodes in a RFC. The model uses the training data to learn relationships of the data that can be used to determine the classification. The validation set is the sample of the data that will be used to generate an unbiased evaluation of the model fitted on the training data set [30]. This set is used during the training phase to evaluate how well a currently fitted model is performing. This data is not used for optimizing the parameters values, it is only used to evaluate a model and to decide which model performs best. The test set is the sample of the data that will be used for an unbiased evaluation of the final model [30]. This data is only used after the training of the model has finished. The model provides a prediction of the instances in the test set based on the final model. The predictions can be used to evaluate how well the final model is performing and often is used to compare different models, such as a RFC with a RNN, to decide which model is most suited in the current setup.

Using a training, a validation and a test set is important to prevent overfitting [23]. In the case that all data is used to train a model, it might result in the model learning relations from the data that are not really occurring in the data. If the model is than used on new data, it might provide incorrect predictions. By using a separate training, validation and test set, this issue is already spotted while developing the model and parameters may be tuned to prevent overfitting. In existing literature no general approach on splitting the data in a training, a validation and a test set exist. Several approaches have been researched and argued why these splitting methods can be used under realistic scenarios [22, 28, 32]. In the current research two methods are used to create a training, a validation and a test set. This step of the framework is related to research subquestion three and four. The third subquestion investigated how the result of the prediction is affected by the creation of the training, the validation and the test set. Therefore, creating these sets based on different methods, will contribute to this subquestion. The fourth subquestion is related to the effect of the prediction quality by applying various windowing techniques. One of the methods to create the training, the validation and the test set is related to windowing techniques. Therefore, this part of the framework also contributed to the fourth subquestion. The first method splits all available traces in the complete log in three subsets defining the training, the validation and the test set. In this case, it is common to use 70% of the data as training data, 10% as validation data and 20% as test data [8]. The training, the validation and the test set can be created by randomly selecting instances from the complete data set. If we look at the time that the trace has been executed, randomly selecting instances for the training, the validation and the test set might result in training on instances that have taken place on a later moment in time than the instances on which the model is tested. Training on instances that occur on a later moment might result in leakage of information and that might provide false insights in the quality of the model [32]. To prevent this, the sets can be created in a time based manner. Therefore, the first 70% of the set chronologically seen in time is used as training sample, then next 10% is the sample of the validation set and the last 20% is the test sample. If the model is used for predicting on new instances, it will be the case that chronologically seen in time the traces have taken place later in time, than the traces on which it has been trained. Therefore, a model that is also tested on the traces that occur on a later moment than the traces on which it is trained, will provide a more realistic view of the results [22]. The second method uses a window of the complete set and only uses traces executed in this window to train and test the model. In this case, a start date and end date of the window is defined. The training and the validation sets are composed the traces that are completed in this window. The test set is constructed of the set of traces of which the proposition defining the decision moment is satisfied in this time window, but that are not yet completed. Using only a window of the data might be useful to investigate whether all historical available data should be used to train a model [27]. As time evolves, new data of the same process will become available. Therefore, it is useful to update a model over time to the new data to ensure up to date predictions. However, training a model with more data is more time consuming. It is worth to investigate whether all historical data are needed or if only the most recent data are sufficient to train a model that provides accurate predictions. Next, pseudo code for both methods of creating a training, a validation and a test set is discussed.

Algorithm 3 explains how the training, the validation and the test set are created for the sets that are created time based. As input it needs the complete event log $\mathcal{L}$ and the list $DecT$ containing the time of the decision moment for each trace and the sizes of the training, the validation and the test set. The sizes of the three sets should be provided by a value between 0 and 1 defining the partition of the set with respect to the complete log. The sizes of the three sets together should sum up to 1, such that each instance of the log is contained in exactly one of the subsets. First $DecT$ will be sorted, such that the case identifiers in the list are ordered chronologically in time based on the decision moment of the traces. Lines 2, 3 and 4 create lists containing the case identifiers for the three sets. $\mathcal{L}_{train}$ contains the first percentage of traces defined by TrS as provided as input, $\mathcal{L}_{val}$ contains the next part defined by VaS and $\mathcal{L}_{test}$ contains the last part defined by TeS. Lines 5, 6 and 7 take the subset of the log $\mathcal{L}$ defining the training, the validation and the test set respectively. On Line 8 the three sets are returned by the algorithm.

Algorithm 3: Create training, validation and test log

```
Input: Event log $\mathcal{L}$, list $DecT$, fraction of training set $TrS$, fraction of validation
       set $VaS$, fraction of test set $TeS$
Output: Three logs $\mathcal{L}_{train}$, $\mathcal{L}_{val}$, $\mathcal{L}_{test}$ defining the training, the validation and
        the test log respectively
1 Sort $DecT$ on the chronologically on time.
2 List $DecT_{train}$ = $DecT$[0: TrS * length($DecT$)]
3 List $DecT_{val}$ = $DecT$[TrS * length($DecT$): VaS * length($DecT$)]
4 List $DecT_{test}$ = $DecT$[(1-TeS) * length($DecT$):]
5 $\mathcal{L}_{train} = \mathcal{L}$ where case identifier $\in DecT_{train}$
6 $\mathcal{L}_{val} = \mathcal{L}$ where case identifier $\in DecT_{val}$
7 $\mathcal{L}_{test} = \mathcal{L}$ where case identifier $\in DecT_{test}$
8 return $\mathcal{L}_{train}, \mathcal{L}_{val}, \mathcal{L}_{test}$
```

Algorithm 4 explains how the training, the validation and the test set are created for a time window. The input to the algorithm is the event log $\mathcal{L}$ and the start and end date and time of the window. First, two empty lists are created. List trainAndValInstances to store the trace identifiers of the traces that belong to the training or validation set. List testInstances to store the trace identifiers of the traces that belong to the test set. Then the algorithm loops over all traces in event log $\mathcal{L}$. If the trace completes within the time frame provided as input, the trace identifier is added to trainAndValInstances. Else, if the decision moment of the current trace is within the timeframe, but the trace has not yet been completed, the trace identifier is added to testInstances. After looping over all traces, the final training and validation sets are created by splitting trainAndValInstances in two sets. The final training, validation and test logs are created by selecting the traces belonging to the trace identifiers contained in the training, the validation and the test list respectively. The algorithm completes by returning the training, the validation and the test sets.

Algorithm 4: Create training, validation and test log using a window

```
Input: Event log $\mathcal{L}$, start date and time of the window $startTime$ and the end
       date and time of the window $endTime$
Output: Three logs $\mathcal{L}_{train}$, $\mathcal{L}_{val}$, $\mathcal{L}_{test}$ defining the training, the validation and
        the test log respectively
1  List trainAndValInstances = []
2  List testInstances = []
3  for trace $t \in \mathcal{L}$ do
4      completeTime = time of last event in $t$
5      decisionTime = time of decision moment in $t$
6      if $startTime \leq completeTime \leq endTime$ then
7          add trace identifier of $t$ to list trainAndValInstances
8      else
9          if $startTime \leq completeTime \leq endTime$ then
10             add trace identifier of $t$ to list testInstances
11         end if
12     end if
13 end for
14 List trainInstances = 85% of trace identifiers of trainAndValInstances with first
     decision times
15 List valInstances = 15% of trace identifiers of trainAndValInstances with the last
     decision times
16 $\mathcal{L}_{train} = \mathcal{L}$ where case identifier $\in$ trainInstances
17 $\mathcal{L}_{val} = \mathcal{L}$ where case identifier $\in$ valInstances
18 $\mathcal{L}_{test} = \mathcal{L}$ where case identifier $\in$ testInstances
19 return $\mathcal{L}_{train}, \mathcal{L}_{val}, \mathcal{L}_{test}$
```

4.4 Data balancing

A sample of the data is a subset of the data, of which it is assumed that it represents the larger group. Sampling might be used to create a balanced data set from an unbalanced data set [11]. Consider spam detection in an email server. Assume that of the complete data set only 1% of the emails is spam and the other 99% of the emails are not. In this case the data is unbalanced, training a classifier model on this data will most likely create a biased classifier towards the majority group. Therefore, it is likely that the classifier has a higher misclassification rate on the minority group. However, it is most interesting to classify these instances correctly, as the minority group is the severe group to get notified about. By sampling the training data set it can be adapted to a representative set, such that the distribution of the emails considered as spam and not spam are 50/50. Such a sampled data set may be used to train an unbiased classifier. This part of the framework contributes to the third research subquestion. The third subquestion is about the affect on the prediction result by the creation of the training, the validation and the test data. This step provides insights in the effect of creating the training data set, as it considers undersampling and oversampling techniques. Therefore, it contributes to answering he third subquestion of this research.

Different techniques exist to sample data. It can be chosen to either undersample the majority class or oversample the minority class. The method of undersampling the majority class entails removing observations of the majority class, such that the number of observations is reduced. This may cause the deletion of important information. Several sampling techniques exist for undersampling data, examples are random undersampling, k-means centroid clustering and Tomek links [11]. On the other hand, oversampling the minority class entails adding copies of existing traces with the minority class outcome. This method might be used if the data set is small and the number of samples is too much reduced when using undersampling. Several sampling techniques exist for oversampling, examples are random oversampling and SMOTE [11]. For both sampling techniques random and non-random methods exist. Random sampling techniques are most easily implemented, as the samples are randomly selected without an underlying heuristic. The main limitation of random selecting samples is that it might result in an adaption of the underlying data distribution. Therefore, it might be the case that the data is not representative anymore [11]. To overcome the limitations of random sampling, it is advisable to use a sample technique that is based on a heuristic. One can also choose to use a combination of oversampling and undersampling to balance the data set.

Data sampling will only be applied on the training data set. The validation and test sets are not sampled such that these set still contain the original and complete set of processes as it is executed by customers. If the validation and test set are sampled as well, the results reported will provide a false representation of the real world [11]. In that case, the model only provides information on the sampled data and not on future real world data. If the model will be used for predictions at a later moment, the predictions might be incorrect in a different number of cases than expected. On the other hand, the training set may be adapted. By balancing the training set, importance is given to the minority class and this will result in a model that is more likely to indicate which instances belong to the minority class.

Algorithm 5 shows how the training set can be undersampled. This method uses the ratio of the occurrence of certain trace variants. The ratio of the occurrences in the original log is equal to the ratio in the undersampled log. As input the algorithm needs the training log $\mathcal{L}_t$ that only contains traces of the majority class and the number of traces $numbTraces$ that should be in the undersampled set. The for loop ranging over Lines 6

Algorithm 5: Undersample training log

```
Input: Training event log with only traces of the majority class $\mathcal{L}_t$, integer
       $numbTraces$ indicting the number of traces that should be kept
Output: Undersampled training event log $\mathcal{L}_{tu}$
1  List orgTraces = [], unique trace ids in $\mathcal{L}_t$
2  List traceVar = [][]
3  List traceIDsSample = []
4  Log $\mathcal{L}_{tu}$ = {}
5  for trace $t \in \mathcal{L}_t$ do
6      if $t$ contained in $traceVar$ then
7          Add traceID of $t$ to second column of the current variant
8      else
9          Add $t$ to the list with in second column the traceID of $t$
10     end if
11 end for
12 for variant $\in$ traceVar do
13     Add $floor$(number of traceID for current variant $/(orgTraces/numbTraces)$)
         traceIDs to traceIDsSample
14 end for
15 Take $length$ traceIDsSample $-numbTraces$ trace ids of the tracesID that are not
     yet taken.
16 for traceID $\in$ traceIDsSample do
17     At trace of traceID to $\mathcal{L}_{tu}$
18 end for
19 return $\mathcal{L}_{tu}$
```

until 11 creates a list of all trace variants. Together with the trace variant $traceVar$ it records the $traceID$ of the traces that are compliant with that $traceVar$. In Line 12 and 13 the algorithm loops over all $traceVar$ and it selects for each variant a number of trace identifiers. The ratio of selected identifiers with respect to the size of the undersampled log is equal to the ratio of the variant in the input event log. This way of selecting traces will result in selecting less traces than the number of traces that is requested. Therefore, the remaining number of traces that are needed are added randomly from the trace identifiers that are not yet selected. The algorithm will return the log containing the traces belonging to the selected trace identifiers.

4.5 Prediction phase of the potential activity

This section elaborates on different machine and deep learning methods to predict the occurrence of the activity of interest. Multiple models are considered such that the result can be compared and it can be researched which model is most suited. The possible methods for prediction are not limited to the models described in this section; therefore, it is also possible to consider another model. As different predictions models are considered in this step, it contributes to answering the second research question. The second research subquestion is related to the influence of the prediction model on the quality of the prediction. By investigation the quality of different models and comparing the results, it provides results to answer the subquestion. First the RFC model is explained in more details. Afterwards and LSTM network is described. Last, the GAN with LSTM networks for suffix prediction is expanded upon.

4.5.1 Random Forest Classifier

In order to train a RFC, the traces first need to be represented as a set of features. These features consist of a set of independent variables and one dependent variable. This set of independent variables should be deduced from the data set and entail the information of the trace that is available at the decision moment. Therefore, the independent variables should only contain information about the events that have taken place prior to the decision moment and the general customer details that are available. Examples of independent variable are the age of a customer, the gender of a customer, the time since the start of the trace up to the decision moment and the number of times the customer has performed a certain activity. All information that can be extracted from the log up to the decision moment might be used as an independent variable. Domain knowledge is useful in determining the independent variables that could be interesting to use. The dependent variable represents the outcome of the activity of interest. It encodes whether the PoAc will occur in the remainder of the trace. This variable can be encoded as 0 if the PoAc does not occur and as 1 if the PoAc occurs. As the dependent variable will only consist of two classes, the decision is a binary decision.

CaseID	Indep. var 1	Indep. var 2	...	Indep. var n	Dependent var
1	value 1	value 2	...	value n	0
2	value n+1	value n+2	...	value 2n	1
3	value 2n+1	value 2n+2	...	value 3n	0
...	...	...	...	...	...

Table 4.1: Example of feature representation as input for a RFC

Table 4.1 shows an example of the feature representation that can be used as input for a RFC. Each row in the table identifies one unique case identifier in the event log. Each trace in the log is translated to the same set of independent variables. Column 2 until $n+1$ contain for each trace the value of the independent variable. The dependent variable represent the occurrence of the PoAc in the suffix of the trace.

Subsequently, the RFC to predict the occurrence of the PoAc can be trained based on the training data set. RFCs are already discussed in Section 2.3.1. During training parameters of the model should be tuned [29], examples are the depth of the trees and the number of decision trees in the forest. The decision nodes are labeled by independent variables and the decisions are based on the possible values of these variables. Leaf nodes contain the class that should be predicted, representing the dependent variable. The quality of the resulting model can be tested with the test data set. The model will only be provided with the independent variables of the instances in the test set and should provide a prediction for each instance. The prediction can be compared with the ground truth value of the dependent variable of the instance to retrieve performance results.

4.5.2 Long-Short-Term-Memory network

This section describes a prediction model that is inspired on the implementation of Kratsch et al. [17] for final outcome prediction. Section 2.4 already provides a description of the work by Kratsch et al. The research by Kratch et al. investigates different ML and DL models on various event logs to gain insights in the properties of an event log that facilitate the use of DL models over the use of ML models. In this research the main goal was to predict the final outcome of running cases. For this purpose, specific preprocessing steps to convert the event log to feature vectors is discussed. The same preprocessing, as proposed by Kratsch et al. will be applied to retrieve feature data for the LSTM model introduced in this section.

The preprocessing applied by Kratsch et al. [17] entailed multiple steps. First, they defined the number x of events which should be considered while creating the feature vector. These x events are the first x events belonging to a trace in the event log. The feature vectors only entail information of the traces that is available up to that moment of the trace. The traces that did not contain at least x events are removed from the log. For the traces that contain more than x events, the x-suffix is not considered while creating feature vectors. For each attribute, on trace level as well as event level, it is checked in what percentage of the traces and events respectively it is occurring. If a certain attribute is contained in less than 1% of the traces or events, it is removed from the log. Otherwise, if a certain trace or event does not contain a certain attribute, but it is occurring in more than 1% of the traces, the attribute is added to the trace or event with a default value. The attributes available in the log will be contained in the feature vectors. One-hot vector encoding is applied to attributes which have categorical values, and the one-hot vector encoding is used in the feature vector. LSTM networks require the same number of attributes in the feature vector at each timestep. With the preprocessing executed, each attribute that is available in at least one event is also available at all other events. Adding the trace attribute variables at each event in the feature vector, creates feature vectors with the same number of features at each timestep. Last, the label of the current trace is assigned to the feature vector. This part of the feature vector is not used while providing predictions. It is only used to compare the prediction with the ground truth and to train the parameters of the model. More details on the original preprocessing can be found in ‘Machine Learning in Business Process Monitoring: A Comparison of Deep Learning and Classical Approaches Used for Outcome Prediction’[17].

This preprocessing is not directly applicable in the current research. The event number of the decision moment may differ from one trace to another, but for each trace the prediction should be provided at the decision moment. Therefore, an additional preprocessing step has to be conducted prior to the preprocessing discussed in the previous paragraph.

For each trace, the number of events prior to the decision moment can be extracted. Furthermore, a number y of events should be defined, which should be used as the preferred prefix length for each trace. Traces containing more than y events up to the decision moment, should be shortened. Only the last y events up to the decision moment should be kept. Traces that have less than y events up to the decision moment should lengthened with artificial events. These events should be added to the start of the trace, such that the number of events up to the decision moment in the trace will get equal to y. The events occurring later than the decision moment, should still be preserved. The suffix of the trace will be used to determine the dependent variable, which indicates whether the PoAc occurs in the suffix.

Algorithm 6: Preprocessing event log for LSTM network model

```
   Input: Event log $\mathcal{L}$, length of trace $l$
   Output: Processed event log $\mathcal{L}$
1  for trace $t \in \mathcal{L}$ do
2      $DecE$ = the event number of the decision moment
3      $EventCount = DecE - l$
4      if $EventCount > 0$ then
5          remove first $EventCount$ event(s) from trace $t$
6      else
7          Add $EventCount$ artificial event(s) to trace $t$
8      end if
9  end for
10 return $\mathcal{L}$
```

Algorithm 6 describes the update of the traces, such that each trace has the same number of events up to the decision moment. As input, the event log $\mathcal{L}$ and a preferred prefix length l are required. The algorithm iterates over each trace contained in the log $\mathcal{L}$. For each trace it determines $DecE$ the event number that defines the decision moment. $DecE$ is compared to l. If $DecE$ is larger than l, the first events are removed, such that the trace prefix gets length l. Otherwise, if l is larger than $DecE$, artificial events are added, such that the trace prefix gets length l.

Table 4.2 shows an example of the feature representation after applying both preprocessing steps. Each row in the table identifies one trace of the event log. All columns except of the last column contain variables representing the trace and event attributes. The last column contains the dependent variable, containing the information on whether the suffix based on the decision moment contains the PoAc. If the suffix contains the PoAc, the value is set to 1, otherwise the value is set to 0.

The feature vectors are used as input to a LSTM network classifier. The classifier is trained with a two-stage learning strategy. The first stage entails hyper-parameter optimization. In this stage multiple hyper-parameter settings are tested by a random parameter search. Random searching for parameters is reasonable, as Bergstra and Bengio have shown that this lead to similar results as testing with all possible combinations of

Trace var 1	...	Event var 1	...	Event var n	Dependent var
value 1	...	value 2	...	value 3	0
value 4	...	value 5	...	value 6	1
value 7	...	value 8	...	value 9	0
...	...	...	...	...	...

Table 4.2: Example of feature representation for the LSTM model

parameters in a more efficient manner [5]. The best model settings are used with a cross-validation to train a final prediction model.

4.5.3 Generative Adversarial Network

The model described in this section is an adaption of the model by Taymouri et al. [35]. Section 2.4 already provides a description of the work by Taymouri et al. Taymouri et al. have developed a GAN with LSTM networks that is used for suffix and remaining time prediction. The generator of the GAN model consists of an encoder-decoder architecture to allow for variable length prefixes and suffixes. The original implementation details can be found in the paper 'A deep adversarial model for suffix and remaining time prediction of event sequences' [35].

First the data preparation phase is explained in more details. The events in the event log should contain a case identifier, activity and timestamp. The activities should be encoded in activity numbers and the timestamp should be provided with a date and time. Next to the original activities, the event log should contain an activity defining the decision moment. This decision moment activity is needed on a later step to define the prefix and suffix of each trace. Additionally, some traces should be shortened. The goal of the current research is to predict whether the PoAc occurs on a later moment than the first occurrence of decision moment. Therefore, as soon as this PoAc occurs in the suffix, the remainder of the suffix is not of interest anymore. For traces in which the PoAc does not occur in the suffix, the complete suffix is of interest. For the traces in which the PoAc occurs in the suffix, the remainder of the suffix executed on a later moment than the event showing that the PoAc occurred should be removed from the trace. Table 4.3 shows an example of the trace representation that could be used as input for suffix creating. For all traces contained in the event log the next elements are captured in the representation: the case identifier, each activity and each timestamp. Each row in the table shows one event captured in the event log.

The implementation described by Taymouri et al. [35] needed some modifications to be applicable for the current research. These modification are regarding the creation of the training, the validation and the test set and the number of prefix and suffixes created for each trace. Taymouri et al. created the training, the validation and the test sets by randomly selecting instances from the complete log for each set. As described in Section 4.3 the creation of the training, the validation and the test set in this research is not random. Therefore, the sets should be defined based on the timestamp of the decision

CaseID	Activity	Timestamp
1	1	2021-05-07 10:12
1	2	2021-05-07 13:15
1	4	2021-05-08 09:00
2	1	2021-05-07 22:04
2	3	2021-05-07 23:30
...	...	...

Table 4.3: Example of trace representation as input for the GAN network for suffix prediction

moment or based on the timeframe. Secondly, Taymouri et al. create multiple prefixes and suffixes per trace. For each trace, a prefix-suffix combination is created for each prefix of length length 2 up to length $|trace| - 1$. Taymouri et al. provide a general prefix-suffix prediction model. In the context of the current research, suffix predictions should only be provided at the decision moment. Therefore, each trace will only consists of one prefix-suffix combination.

Algorithm 7 shows the process of data reading and suffix generation with the GAN model. As input the training, the validation and the test data set are needed with the representation as provided in Table 4.3. For each event, represented by a row, occurring in the data set, the categorical activity is represented by a binary vector via one-hot vector encoding, the timestamp is replaced by the time that is elapsed since the previous event and the current event, defining the duration of the event. This part of the algorithm is represented by Lines 3 till 9. Lines 10 till 13 splits each trace to the correct prefix and suffix. The prefix of a trace is the original trace up to the event defined as the decision moment. The suffix of the trace are the events of the original trace occurring on a later moment than the decision moment. Line 15 defines the creation of the GAN model $\mathcal{M}$. This creation entails the definition of the generator and discriminator model. These are LSTM networks, for which parameters, such as number of layers, and batch size need to be defined. The next step is to train model $\mathcal{M}$ with the training and the validation data frame created from $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ respectively. After training model $\mathcal{M}$, the model can be used to provide new predictions. These predictions are suffixes created by the generator. The prefixes of the test data frame are provided as input, the model will predict the suffix for each prefix. The model $\mathcal{M}$ as well as the predicted suffixes on the test set are returned by the algorithm. The model can later be used on different data to provide more predictions, the predicted suffix can be compared with the ground truth suffixes of the test set to retrieve performance measures.

4.6 Model comparison

In Section 4.5 three different models for PoAc prediction are discussed. All these models will produce own predictions and the models will provide different predictions for a part

Algorithm 7: GAN suffix generation

```
   Input: Training event log $\mathcal{L}_{train}$, validation event log $\mathcal{L}_{val}$, test event log $\mathcal{L}_{test}$
   Output: GAN model $\mathcal{M}$, suffixes for the test set $suffix\_test$
 1 Create empty dataframes $dataTrain$, $dataVal$, $dataTest$
 2 for $data\ set \in [\mathcal{L}_{train}, \mathcal{L}_{val}, \mathcal{L}_{test}]$ do
 3     for $trace\ t \in dataset$ do
 4         for $event\ e \in trace$ do
 5             Create one-hot vector for $e$
 6             Define duration time of current event
 7             Add one-hot vector and duration time to row in dataframe of current
                 data set
 8         end for
 9     end for
10     for $trace\ t \in dataset$ do
11         create the prefix of $t$ based on the decision moment
12         create the suffix of $t$ based on the decision moment
13     end for
14 end for
15 Create GAN model $\mathcal{M}$
16 Train $\mathcal{M}$ with $dataTrain$ and $dataVal$ data
17 $suffix\_test$ = predicition of $dataTest$ on $\mathcal{M}$
18 return $\mathcal{M}$, $suffix\_test$
```

of the instances. The next step is to evaluate the performance of each classifier to judge on the trustworthiness of the classifiers and to compare the different models. Comparing different predictions models contributes to answering the second research subquestion. That subquestion is concerned with the influence of the prediction model on the quality of the prediction. By comparing different models, information is gathered on the influence of the prediction model on the prediction. To determine the quality of the models, the predictions should be compared with the ground truth. Afterwards, a comparison between the models can be performed to decide which model is most preferred to use.

The first step in model comparison is to determine the quality of each model separately. For each model, the test set is reserved as unseen data to retrieve information on model quality. For this test set it is known whether the PoAc will occur on a later moment than the decision moment and the needed representations for each model are available. Therefore, we can use the generated model together with the test set to retrieve predictions on the occurrence of the PoAc. The predictions of each model can be compared to the ground truth to get insight in the trustworthiness of the model. The RFC and LSTM network provide predictions on the occurrence of the PoAc by default. For the GAN model, suffixes are provided and it should first be extracted whether the PoAc is occurring in the suffix, to decide on the quality of the model and to be able to compare the results to the other two models. The quality of each model will be accessed by the F1-score or the recall score. Section 5.2 provides more details on the performance measures. Depending on the research field and goal of the research, either the F1-score or recall and F1-score will be most interesting to consider. For research fields at which it is important to detect all occurrences of the potential activity, the recall score is very important. This can be the case if we want to detect whether a fraudulent activity is expected to happen. In this case the recall and F1-score should be considered. However, if it is just as important to only predict a PoAc to happen when it really happens as selecting all cases of the PoAc, only the F1-score should be considered.

In addition to deciding the quality of each model separately, models should be compared against each other. This comparison can be conducted with the same F1 and recall scores. Section 5.2 provides more details on the performance measures. Generally, a higher score implies that the model is outperforming the other models [1]. This comparison is meaningful to decide on the model that can best be used in future cases. After training a model, the goal is to predict for new cases, as soon as the decision moment property holds, whether the PoAc will occur. Predictions should be as reliable as possible in such cases; therefore, it is important to generate predictions with the model that is expected to be most trustworthy.

4.7 Detailing the predictions

After applying HIAP once, the model can be re-iterated to retrieve more detailed predictions. This is visualized in Figure 4.1 by the arrow labeled with '7. Detailing the predictions'. During a new iteration the same research is applied on a different decision moment in the process or on a more into depth PoAc. Applying the prediction of the occurrence of the PoAc on different moments in the trace, might provide new insights and trade-offs. Predictions conducted early in the process might be interesting, to provide early insights in the expectation on whether a customer will perform the PoAc and allows for early adaption of the customer journey. However, later predictions might be more accurate and prevent unnecessary adaptions of the process. Therefore, it might be interesting to conduct predictions on multiple decision moments of the same process to retrieve more in depth insights. Alternatively, re-iterating can be used for a more into depth research. For example, the first iteration can predict whether a customer is expected to request information about a product in the shop, independent of the exact product. A second iteration could be used to get insight in the type of the product. Thus for the customers who are predicted to ask more information about a product, it might be predicted whether this product will be a mobile phone or a phone accessory. Re-iterating the HIAP can provide more insight in the process under investigation. During the re-iteration, either all steps or all steps except the data harmonization step have to be applied again. When a new decision moments is chosen in the same process, the data harmonization can be skipped. For the purpose of a more detailed PoAc the data harmonization should again be conducted. In this case it might be that the research should not contain all completed cases, but only completed cases in which a certain activity had occurred. Therefore, the cleaning might remove more customer journeys from the log. The other steps should always be re-executed. The decision moment and potential activity should again be defined. New training validation and test sets should be created and balanced. The different prediction models should be retrained. And the new predictions should again be compared with the ground truth for model comparison. The performance of the first iteration can be compared to the performance of the new detailed iteration. For example, if another decision moment is chosen, insight can be gathered in the trade-of between the earliness of the prediction and the quality of the prediction by comparing the different iterations of HIAP.

4.8 Future model use

Section 4.6 discussed how to determine which model was performing best. This model can be used for future instance predictions. After training a model, the model can be stored, such that the model can be used for future predictions of the activity of interest. New customer journeys are journeys executed by the customer on a later moment. At the moment that the proposition defining the decision moment holds, the new journey can be represented with the same feature representation as the testing data was represented. For this purpose, the customer journey up to that moment should be mapped to the representation as provided in Sections 4.5.1, 4.5.2 and 4.5.3 for the RFC, LSTM and GAN model respectively. The same mapping to create the features should be used as was originally used to create the training, the validation and the test data. This ensures that the same features of the trace are captured in the feature vector. For the new customer journeys, the log should not be separated in a training, a validation and a test log as the model has already been trained and will only be used to provide predictions. The predictions present insights in whether the customer is expected to eventually interact with the PoAc.

Chapter 5

Experimental evaluation

This chapter describes the application of the HIAP research on two data sets. The first data set is provided by VGZ, a Dutch health insurer. The second data set is the BPI 2012 benchmark data set of a loan application process attained from the 4TU Centre for Research data [10]. The BPI 2012 data set is selected to show that the HIAP is not only applicable to specific domains, but can be applied to a wider range of research. Both data sets are described in Section 5.1. Next, Section 5.2 describes how the application of the HIAP can be evaluated. The baseline model to which the results can be compared is introduced in Section 5.3. Section 5.4 explains how the customer journey can be pre-processed and analyzed using process mining techniques. Then, Section 5.5 describes how classification models are implemented for the HIAP and analyzes the quality of the different models. Additionally, Section 5.6 evaluates the effect of creating the training, the validation and the test set in a time based manner or by a random split. The performance measures are compared for both methods of creating the training, the validation and the test set. Section 5.7 discusses methods to retrain the model over time. It discusses the difference of only using the most recent data to train a model and using all historical available data to retrain a model. Last, Section 5.8 finishes the experimental evaluation by summarizing the results of applying HIAP.

5.1 Data sets

To evaluate the performance of the HIAP, it is applied to two data sets. This section introduces the data sets used for this evaluation. First, the description of the Dutch health insurer set is provided and then the BPI 2012 data set is introduced.

5.1.1 Health insurer data set

The health insurer data set is a data set provided by the Dutch health insurer VGZ. The data contains details about the declaration process for customers and has been stored over different sources. This declaration process starts once a customer has payed an invoice for a certain care that the customer has used and wants to get the amount paid refunded from VGZ. To achieve this the customer submits the invoice to the insurer. This invoice will be checked by the insurer and a decision including the reason will be sent to the customer. The declaration process data covers a time period of two months. It consists of around 90,000 customers who have submitted around 147,000 declarations. Next to the activities related to the declaration, information of customers visiting the website and contact VGZ by phone are available. Furthermore, some anonymized customer details are available. For each touchpoint the interaction, timestamp and customer identifier are recorded. Touchpoints related to submitting or retrieving results of a declaration are coupled to a declaration identifier next to the customer identifier. Each touchpoint has its own attributes included in the data. The attributes of a phone call are the subject and customer question. A website visit contains the visited url, browser used by the client and an interaction identifier. Submission of declarations contain channel information over which the declaration has been submitted. Declaration decisions contain the message for the client, type of reimbursement and the category of the decision. The category of the decision is a categorisation of the acceptance and rejection reason.

5.1.2 BPI 2012 challenge data set

The BPI 2012 challenge event log contains data of the application process for a personal loan or overdraft within a Dutch financial institute. Events belong to a sub-process and the complete log contains three sub-processes. The application sub-process is concerned with the handling of the applications. Offer sub-process defining the events belonging to the offers sent to customers for a certain application. Workflow sub-process describing work items for certain applications. The event log covers a time period of 6 months and contains around 13,000 cases and 262,000 events. For each event stored in the event log the event, the lifecycle of the event, the timestamp, the customer identifier and the requested amount are stored. The requested amount specifies the amount of loan or overdraft requested by a customer in the current application. Activities for which an organizational resources is needed, also specify the organizational resource that executed the activity.

5.2 Performance measures

Performance evaluation is crucial to compare different classification models. The prediction of the classifier should be compared to the ground truth of the same data instance. If the prediction matches the ground truth, the prediction is referred to as a true prediction (T). Otherwise, if the prediction is different from the ground truth, the prediction is referred to as a false prediction (F) [1]. In this research, binary decision about the occurrence of an activity of interest are provided. This allows the results to be presented in a 2 by 2 matrix defining the classes and number of predictions in such class. Table 5.1 shows a confusion matrix, denoting the classes. The class for which the ground truth is positive and the prediction is positive defines the truth positives (TP), if the prediction is negative it defines the false negatives (FN). The class for which the ground truth is negative and the prediction is positive defines the false positive (FP) and if the prediction

Ground truth		**Predicted value** Negative	Positive
	Negative	TN	FP
	Positive	FN	TP

Table 5.1: Confusion matrix

is negative it defines the true negatives (TN) [1]. In terms of predicting an activity of interest, the positive class corresponds to the occurrence of the PoAc and the negative class to non-occurrence of the PoAc. Several performance measures can be calculated on the basis of these classes. The most common measure is the accuracy of the predictions [1].

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

The accuracy aggregates all predictions in a single measure, however, using the accuracy measure may provide misleading results. Assume, a test set contains 99 cases with a negative ground truth value and 1 with a positive value. A classifier predicting the negative class in all cases will reach a high accuracy of 99%. Nevertheless, in such cases it is interesting to correctly classify the positive class and if this is never predicted, the classifier model should not be evaluated as performing well.

Especially for prediction models with imbalanced classes and the minority class being the class of interest to predict class-wise measures are needed [1]. Such measures are precision, recall and F1-score. For all three measures, the resulting value will be between 0 and 1, where values closer to 1 indicate that the model is performing good.

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

$$F1\text{-}score = 2 \times \frac{precision \times recall}{precision + recall} \tag{5.4}$$

The precision measure (Equation 5.2) indicates how many of the positive predicted instances are positive based on the ground truth as well [1]. A higher precision implies that there are fewer cases that are wrongly predicted as positive. The precision measure might be a good measure in the case that mails should be classified as spam. In the context of spam detection it is important that no important non-spam email is predicted to be spam and therefore no FP predictions should occur. In contrast, recall (Equation 5.3) indicates how many of the ground truth positive instances are predicted to be positive [1]. A higher recall measure implies that more positive classes have been detected. The recall measure might be a good measure in the case of fraud detection. In the context of fraud detection it is important that all cases that are fraud are detected and that no cases are missed. However, recall should not be used as the only performing measure. A prediction model that is predicting the positive class for all instances will achieve a recall of 1, however, this will also provide FP predictions. It can be the case that a classifier model retrieves a high recall and a low precision, it might also be that a classifier model retrieves a low recall and a high precision. In the best case, the precision and recall scores are both high. Recall and precision can also be combined to a single measure [1]. This measure is the F1-score (Equation 5.4). The F1-score seeks a balance between precision and recall and is the harmonic mean of the two measures. If either the precision or recall score is low, then the F1-score is also low. Only if both precision and recall are high, the F1-score is high as well.

Depending on the purpose that the classifier model has to achieve, the most important performance measure can be determined. If both recall and precision are important, it is best to consider the F1-score. If only the precision is of importance, this score can be used to determine the quality of a model and to compare different models. In the case that recall is important, as well recall as the F1-score should be considered. Scoring a recall score of 1 is possible by always predicting the positive class. This will result in a model that is able to detect all positive classes, but it might not be the case that the model is performing well. Therefore, the recall scoring should not be used as the only measure to decide on the performance of a model. However, if recall is important, it can be considered as one of the measures to determine the quality of a model. In terms of predicting the occurrence of the PoAc, it is important to predict the occurrence correctly in the positive ground truth class. This indicates that the recall score is an important measure to consider. However, this measure should not be used solely; therefore, the F1-score as well as the recall score should be considered while determining the quality of the prediction model. During model comparison all four measures will be reported, with the recall and F1-score being the central performance measure to consider.

5.3 Baseline model

To reason about the quality and added value of the HIAP, a baseline model should be defined in addition to performance measures. The chosen baseline model is a random predictor, which may be assumed to be the baseline for prediction performance. It should be the case that the HIAP is performing better than the baseline model, as it otherwise would mean that random guessing would provide better predictions.

A random predictor is based on randomness and therefore simple to implement and explain. The input for a random predictor is the distribution of the traces containing and not containing the activity of interest in the suffix related to the decision moment. To test the random predictor an artificial data set is created. The artificial occurrence of the PoAc will be equal to the distribution in the original data. The random baseline model does not use any input information of traces to assign predictions to each instance in the set. The random predictor, assigns a random number between 0 and 1 to each instance in the artificial data set. The prediction of the random generator is determined by rounding the random number. Therefore, a random number ≤ 0.5 will be assigned a 0 indicating that the prediction is equal to no occurrence of the activity of interest. Otherwise, the assigned prediction is 1 indicating that it is expected that for this instance the activity of interest will occur. Table 5.2 shows an example of how a random baseline model could look. The first column shows the artificial outcome of the occurrence of the activity of interest. Then for each instance in the artificial data set a random number is generated and the prediction of the random predictor model is a result of this random prediction. The performance measures of the model can be calculated by using the values in the column 'Occurrence of the PoAc' as ground truth values and the 'Prediction' column as predicted values. This process of randomly generating predictions is repeated 1000 times. For each

Occurrence of the PoAc	Random number	Prediction
0	0.05	0
0	0.91	1
...	...	...
1	0.79	1
1	0.78	1

Table 5.2: Example of a random predictor result

iteration the accuracy, precision, recall and F1-score is determined. The final performance of the random predictor is determined by taking the average of each measure over the 1000 iterations. The random prediction are repeated 1000 times as this will result in values that are normally distributed as proven by the Central Limit Theorem [21]. Using the average of the values that are normally distributed result in a valid representation of the obtained results.

The baseline model for the VGZ data set and the BPI 2012 challenge data set will be discussed in Sections 5.5.1 and 5.5.2 respectively.

5.4 Data set preparation and result of customer journey analysis

This section presents the results of the data harmonization, definition of critical moments and the creation of the training, the validation and the test set for the VGZ data set and BPI 2012 challenge log. Applying those steps on the two data sets demonstrates how customer journey data can be analyzed by the use of process mining techniques. Using process mining techniques in the context of customer journeys demonstrates the steps of the model explained in Sections 4.1 to 4.4. The steps will first be applied on the VGZ data set in Section 5.4.1 and then on the BPI 2012 challenge log in Section 5.4.2.

5.4.1 Health insurer data set

This section shows the application of the data harmonization (section 4.1), the definition of the critical moments (section 4.2), the creation of the training, the validation and the test log (section 4.3) and balancing of the training data (section 4.4) on the health insurer data set.

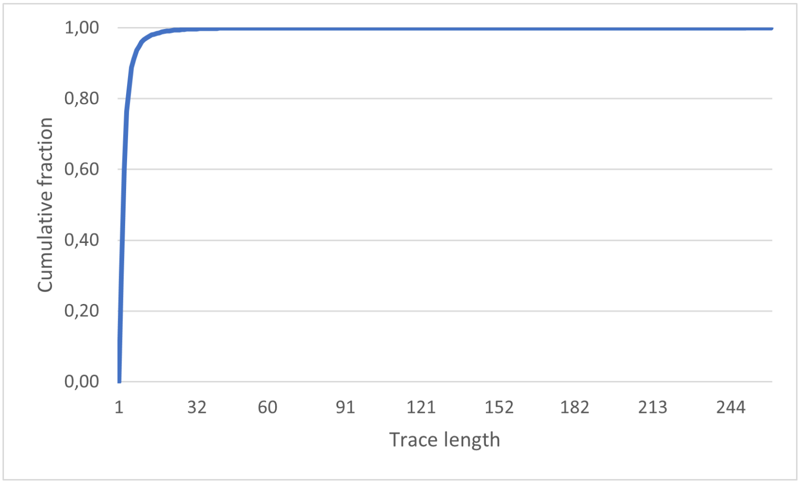


Figure 5.1: Cumulative trace length distribution of the VGZ data set

The first step in applying the methodology described in Chapter 4 is the data harmonization step. The data harmonization step has been performed in collaboration with VGZ, as domain knowledge improves the quality of the events entailed in the resulting event log. Data harmonization involves domain knowledge and is specific for each data set [2]; therefore, no harmonization details specific to the current data set are provided, but only the general steps are explained. The first step is the removal of interaction attributes that are not deemed interesting in the current research. Certain phone calls of which the subject of the call was not related to the declaration process are also removed. Next, interactions of the customer log are transformed to events of the event log. The mapping of touchpoints and touchpoint attributes is constructed with domain knowledge. In the original customer log, each interaction belonged to a certain customer. For the event log it is determined that traces are identified by the declarations. To prevent the loss of all phone calls and website visits, the interactions are transformed to belong to a declaration. Each phone call or website visit that was performed in a time span within 7 days prior to the submission of the declaration and 5 days after retrieving the result of the declaration is mapped to belong to that specific declaration. Using this transformation, also incomplete customer journeys could be removed from the data. Website visits and phone calls prior to the declaration might be missing for declaration that are submitted in the first 6 days of the time frame. Therefore, these customer journey are deemed incomplete and removed from the log. Customer journeys of which the 5 day time span after retrieving the result is not completely available in the customer log are also considered as incomplete. After defining mappings from touchpoints to activities, touchpoint attributes to event attributes and timestamp to timestamp, the customer log is transformed to an event log.

After applying the data harmonization steps the event log consist of 95,457 traces accounting for nearly 400,000 events. Figure 5.1 shows the cumulative distribution of the trace lengths occurring in the event log. It can be observed that most of the traces are relatively short. 95% of the traces have contain less than 10 events.

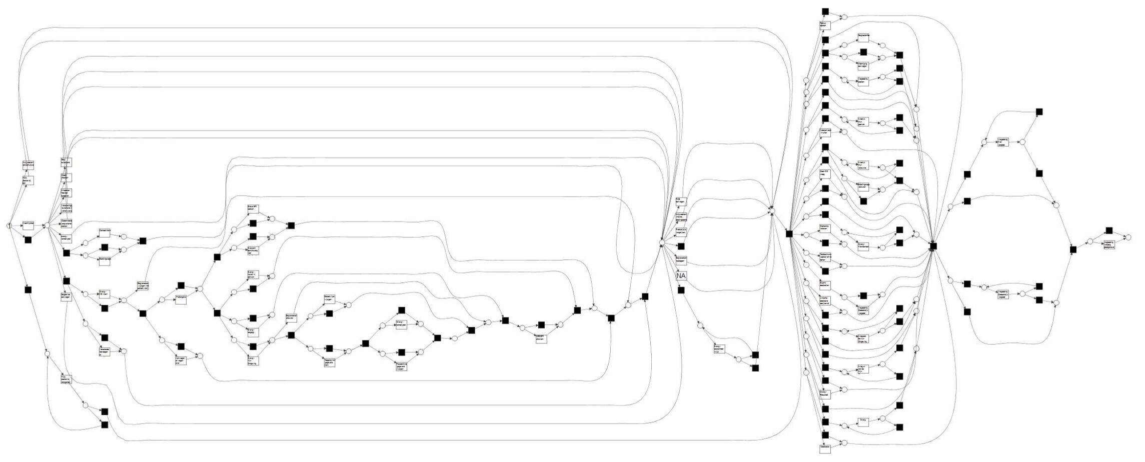


Figure 5.2: Petri net of the traces in the VGZ data set

From the event log process models have been created to retrieve insights in the process. Figure 5.2 shows an attempt to create a petri net, the result is an unstructured petri net. The data contains a lot of different events and a lot of different traces are contained in the event log. Therefore, the resulting petri net is difficult to interpret. To retrieve a more global overview, the events occurring in the event log have been abstracted and the new model is shown in Figure 5.3. The model is shown to give a general feeling of the process, appendix A contains a larger version of the petri net. This petri net shows a lot of tau transitions, showing that multiple events can be skipped or executed multiple times. The events describing the activity of a customer calling to VGZ or visiting the website can be executed at any moment in the trace. As the current process is difficult to examine via a process model, the next steps are mainly based on domain knowledge.

Formerly, based on domain knowledge and the insight gathered in the data harmonization step the critical moment for the process is determined. The goal for VGZ is to determine whether a customer will call VGZ as a follow-up to obtaining the result of the declaration. This activity is often perceived negatively; therefore, VGZ would like to prevent the occurrence of a call. The first step to prevent the call is to know who will call. For this reason, the PoAc for VGZ is defined as a call. It is interesting to predict the call independent of the exact subject. The calls that will be predicted are the calls that occur on a later moment than that the result of a earlier submitted declaration is received. Resulting in the decision moment being the moment that the result of a declaration is sent to the customer. For the events occurring in the trace on a later moment than that the result is received, it should be determined whether the customer calls at least once. If this is the case, the PoAc has a positive outcome. Otherwise the PoAc has a negative result of not occurring in the suffix. In the data harmonization step, only traces that are deemed complete are left in the event log. Traces are deemed complete only if the customer has

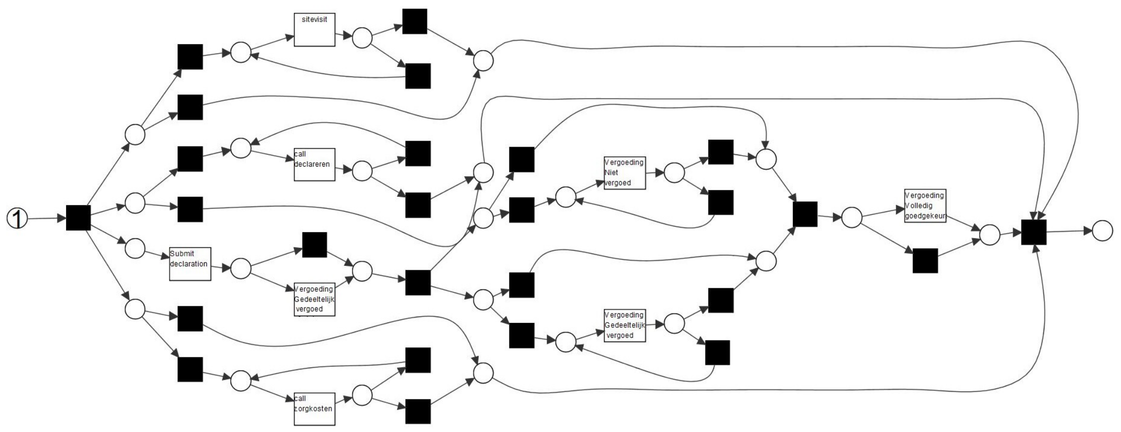

1
sitevisit
call declareren
Submit declaration
Vergoeding Gedeeltelijk vergoed
call zorgkosten
Vergoeding Niet vergoed
Vergoeding Gedeeltelijk vergoed
Vergoeding Volledig goedgekeur

Figure 5.3: Petri net of the traces in the VGZ data set with abstracted events

had a reaction on its filed declaration. Therefore, Algorithm 1 will not remove any traces from the log, but will only insert the decision moment at the correct position in the event log. Figure 5.4 shows the number of traces which contain and do not contain the PoAc in the suffix with respect to the decision moment. It can be seen that the occurrence of the to PoAc is unbalanced. Only 3.5% of the traces in the log contain a call event of the customer on a later moment than receiving the result on a declaration.

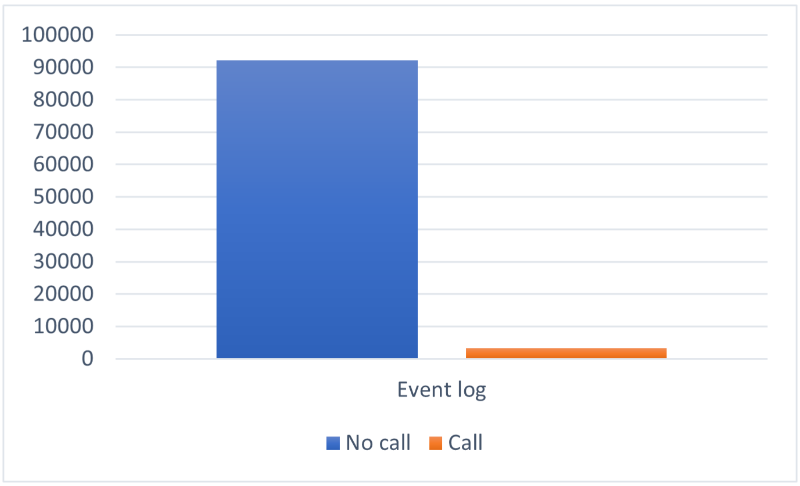


Figure 5.4: Distribution of the potential activity in the VGZ event log

The next step is to split the event log in a training, a validation and a test set. These sets are created by Algorithm 3. The size of the training set is 0.7, the validation set is 0.1 and the test set is 0.2. Therefore, the first 70% of the traces based on the time of the decision moment are contained in the training log, the next 10% are contained in the validation log and the last 20% of the traces define the test log. Figure 5.5 shows

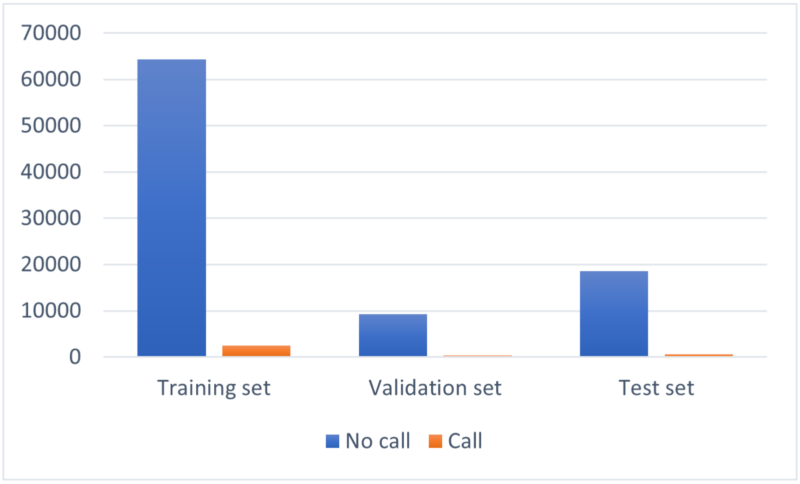


Figure 5.5: Distribution of the potential activity in the VGZ event log over the training, the validation and the test set.

the distribution of the PoAc of the three event logs. For all event logs the number of traces which do not contain an event defining a call is much larger than the number of traces which contain an call. 3.7%, 3.3% and 2.9% of the traces contain the PoAc for the training, the validation and the test set respectively. The distribution of the PoAc is imbalanced. Training data should be balanced prior to training a prediction model. Based on Algorithm 5 the training set will be undersampled, such that the distribution of the occurrence of a call in the suffix of the traces with respect to the decision moment is 50/50. The training set contains 2453 traces in which the customer has called VGZ after the customer has received the result of the filed declaration. Therefore, the group of traces which do not contain a call event is reduced to 2453 traces as well. Undersampling the majority class concludes the preparation of the VGZ event log.

5.4.2 BPI 2012 challenge data set

This section shows the application of the data harmonization (section 4.1), the definition of the critical moments (section 4.2), the creation of the training, the validation and the test log (section 4.3) and balancing of the training data (section 4.4) for the BPI 2012 challenge data set.

The initial step to apply on the BPI 2012 challenge data set is the data harmonization step. The BPI 2012 data set is already provided as an event log and the data is already provided in one data source. Even though the process is provided as an event log, it describes the customer journey of customers applying for a loan. The transformation of elements from customer log to event log could be omitted and only the cleaning step needs to be executed. Cleaning entails the removal of events and removal of traces that are deemed unimportant. For the current research, only events with the attribute value of 'complete' for the life cycle are considered; therefore, events with the life cycle 'start' or 'schedule' are removed from the event log. Additionally, only complete traces are of interest and traces that are not completed in the available time span should be removed. Traces are completed as either the application is approved, cancelled or declined. Each trace that does not contain an activity describing that the application is approved, cancelled or declined is removed from the log.

Applying the aforementioned cleaning step, the event log contains traces of nearly 12,700 customers, together accounting for over 156,000 events. The cumulative trace

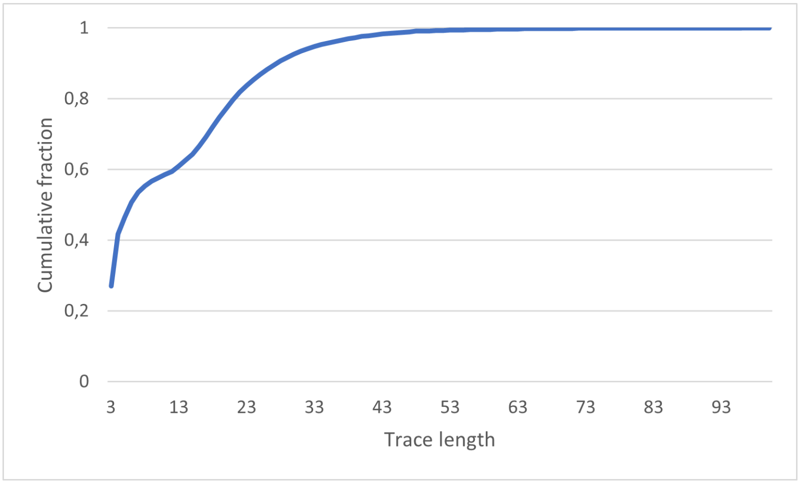


Figure 5.6: Cumulative trace length distribution of the BPI 2012 data set

length of the traces in the cleaned BPI 2012 event log is shown in Figure 5.6. It can be observed that most traces have a length of at most 9 events or consists of 15 to 24 events, as at these lengths the graph is showing the steepest growth. Figure 5.7 shows the process model of the traces in the cleaned BPI 2012 event log. This model can be used to retrieve insights in the customer journeys captured in the event log. The process starts with a customer submitting an application. This initial application is either pre-accepted or declined. After an application has been declined, the trace might already be completed. For not yet completed traces, the next steps are workflow items to retrieve additional required information. Afterwards, an offer might be created and sent to the customer. One trace can contain multiple offers; therefore, the creation and sending of offers can be executed multiple times. After sending an offer, again a workflow item takes place to call the customer who received an offer. An offer is either cancelled or sent back indicating that the customer has responded to an offer.

The critical moments for the BPI 2012 challenge log are determined based on the insights that can be extracted from the process model, since domain experts cannot be consulted. A new process in the log is initiated if a customer request for a loan, in that case the Dutch financial institute determines whether an offer will be sent to the customer. In order to determine whether an offer will be sent human resources are needed to complete the application and to create an offer. If it is known early enough whether an offer will be sent, the resources could be used only for cases in which indeed an offer

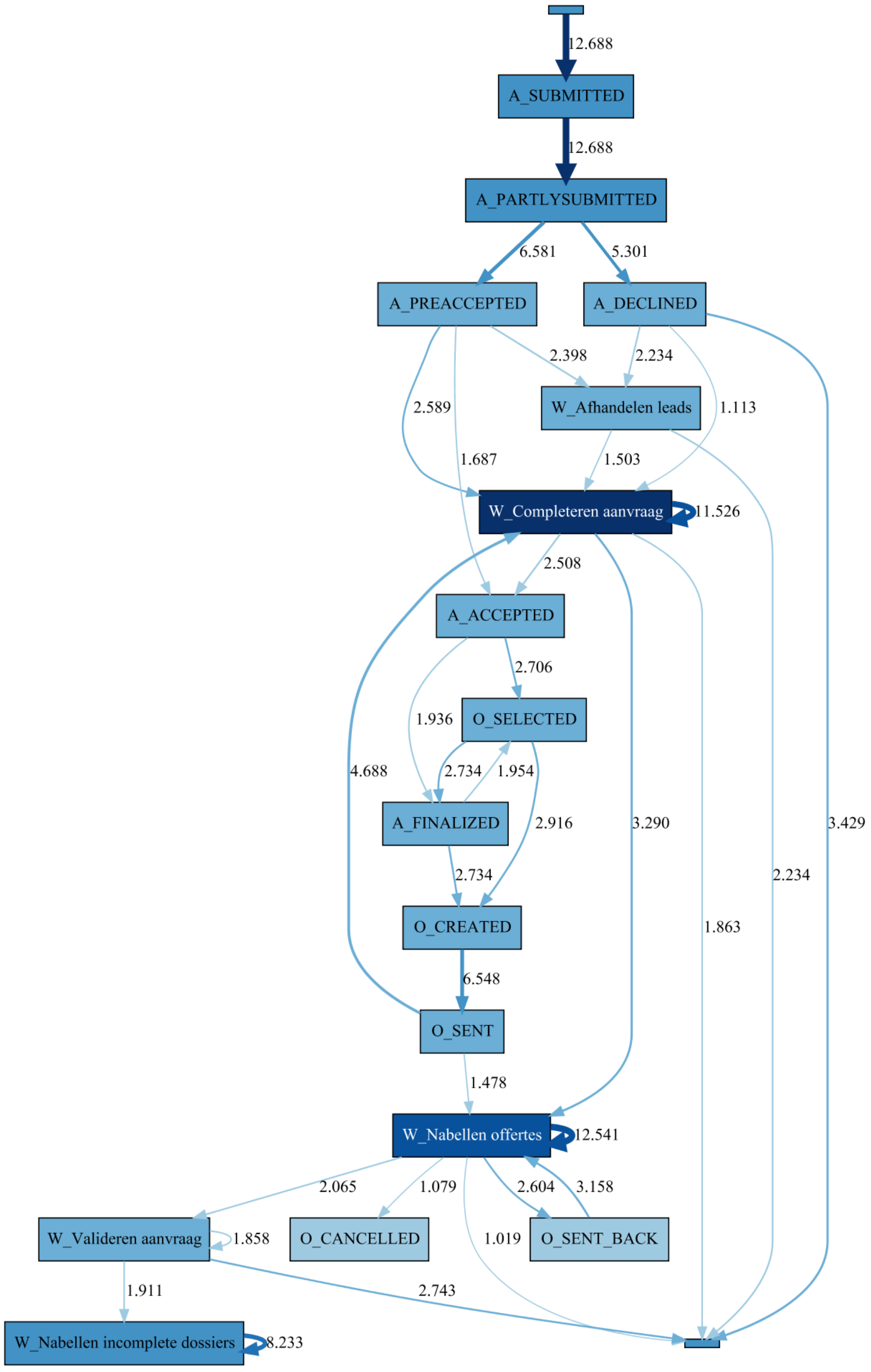


Figure 5.7: Directly follow graph of the traces in the BPI 2012 data set

will be provided to the customer. Therefore, the PoAc to predict whether it will occur is the activity 'O_SENT'. At the activities of 'W_Completeren_aanvraag' (Complete application) and 'A_PREACCEPTED' the remainder of the process can still contain the activity 'O_SENT', but the process might also finish without the activity 'O_SENT'. Accordingly, two decision moments are defined, the first decision moment being the moment at which 'A_PREACCEPTED' occurs and the second decision moment being the moment at which 'W_Completeren_aanvraag' occurs. For both decision moments the same PoAc will be predicted. Thus, at the decision moment it should be predicted whether at a later moment in the trace than the decision moment the activity 'O_SENT' will occur. A positive outcome indicates that it is predicted that an offer will be sent, otherwise the prediction has a negative outcome of not occurring in the suffix. For the prediction task on the decision moment of 'A_PREACCEPTED' only traces in which the activity 'A_PREACCEPTED' occurs are considered. By applying Algorithm 1, 5720 traces are removed, such that the returned log has 6968 traces. Algorithm 1 is also used to create an event log for the decision moment which is defined by the activity 'W_Completeren_aanvraag'. The algorithm again returns an event log consisting of 6968 traces. The distribution of the occurrence of the PoAc for

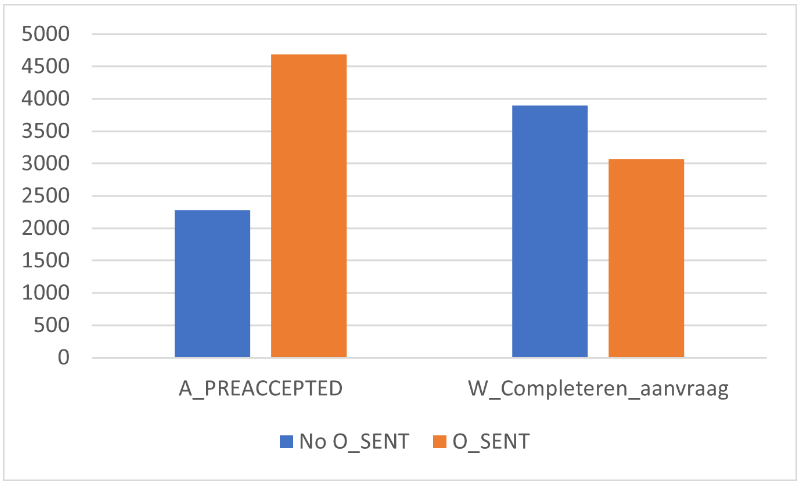
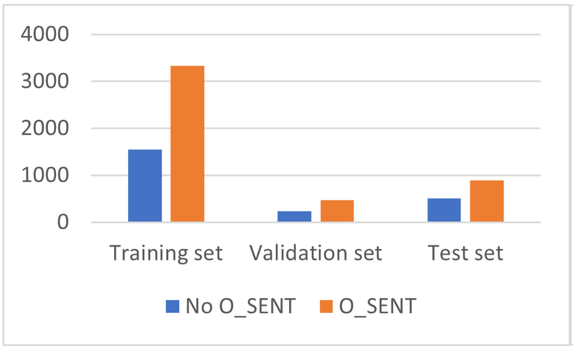


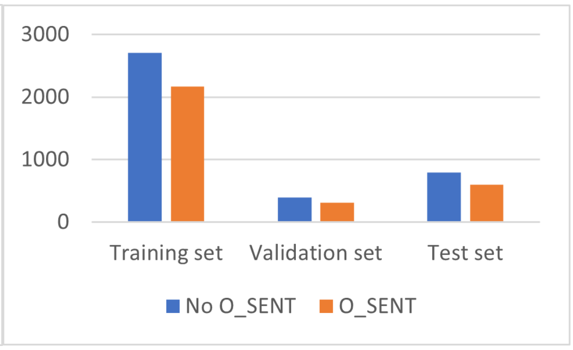
Figure 5.8: Distribution of the potential activity in the BPI 2012 challenge log

both sets is shown in Figure 5.8. The left part of the Figure 5.8 shows the distribution of 'O_SENT' for the log in which the decision moment is defined by 'A_PREACCEPTED', while the right part show the distribution for the log of the decision moment defined by 'W_Completeren_aanvraag'. More often an offer is sent to the customer in the remainder of the trace for the log at which the decisions moment is defined by 'A_PREACCEPTED' than 'W_Completeren_aanvraag'. The occurrence of 'O_SENT' is $67,2\%$ and $44,1\%$ for the log defined by 'A_PREACCEPTED' and 'W_Completeren_aanvraag' respectively.

The next step in the data set preparation is the creation of the training, the validation and the test set. This step is performed for both data sets separately. For both data sets Algorithm 3 is used to create the training, the validation and the test set. The sizes of the training, the validation and the test set are 0.7, 0.1 and 0.2 respectively. Figure 5.9 shows the distribution of the PoAc over the training, the validation and the test set of the two data sets. The distribution over the training, the validation and the test set of the occurrence of 'O_SENT' of the data set defined by the decision moment 'A_PREACCEPTED' is shown in Figure 5.9a. Figure 5.9b shows the distribution over the training, the validation and the test set of the occurrence of 'O_SENT' of the data set defined by the decision moment 'W_Completeren_aanvraag'. For all subsets the distributions of the occurrence of an offer that has been sent to a customer is almost equal to the distribution in the complete set. Prior to model training both training sets will be balanced, such that the distribution of the occurrence of 'O_SENT' is 50/50 in both sets. The part of the training set in which the suffix contains the activity 'O_SENT' indicating that an offer is sent to the customer of the log defined by 'A_PREACCEPTED' should be undersampled by Algorithm 5. For the training set of the event log defined by the decision moment 'W_Completeren_aanvraag' the subset of traces which do not contain the activity 'O_SENT' should be undersampled by Algorithm 5.



(a) Decision moment 'A_PREACCEPTED'



(b) Decision moment 'W_Completeren_aanvraag'

Figure 5.9: Distribution of the potential activity in the BPI 2012 challenge log over the training, validation and test set.

5.5 Result of the prediction phase and model comparison

This section presents the results of the prediction phase of the PoAc and the model comparison. It demonstrates the influence of the prediction model on the quality of the predictions. Section 5.5.1 discusses the RFC, LSTM and GAN model implementation on the VGZ data set. The prediction results are compared with each other and with respect to the random predictor. Section 5.5.2 describes the implementation and result comparison of the RFC, LSTM, GAN model and random predictor on the BPI 2012 challenge log on decision moment 'A_PREACCEPTED' and 'W_Completeren_aanvraag'.

5.5.1 Health insurer data set

This section shows how the RFC, LSTM network and GAN model are applied on the preprocessed VGZ data set. The data is preprocessed as discussed in Section 5.4.1. The models will be trained to predict whether a customer is expected to call VGZ in the remainder of its customer journey. The prediction should be provided on the moment that the customer receives its result on an earlier submitted declaration. Furthermore, the results of the random predictor are retrieved and the comparison of the different prediction models is provided.

Random Forest Classifier

Prior to training and testing the RFC, the traces have to be converted to a feature representation. The information stored in the independent variables of features may only be extracted from the trace available up to the decision moment. No information may be used of events occurring on a later moment. The customer details that are known, may also be used in the creation of the independent variables. No general feature construction is available, as the features that can be constructed are dependent on the underlying data set [8]. Therefore, the exact set of features extracted from the data set is not discussed. The same set of features should be extracted for the traces in the training set as the traces in the test set. In this step the dependent variable should also be defined for each trace. List id_α and id_β (retrieved from Algorithm 2) contain the identifiers of the traces of which the customers have called to VGZ and identifiers of the traces in which no call occurred respectively. These sets are used to determine the dependent decision variable. The dependent variable of the traces of which the identifier is contained in id_α is equal to 1, as for these traces the customer has called to VGZ. While the dependent variable of trace identifiers contained in id_β is equal to 0.

The feature representations are used to train and test a RFC. During training of the model, the hyper parameters of the model are tuned with a randomized grid search over possible parameter settings. Afterwards, the RFC is tested with the instances contained in the test set. The performance of the RFC is determined by comparing the prediction provided by the RFC with the ground truth. Performance measure results are shown in Figure 5.11. The blue bars show the performance measures of the RFC. The results will be discussed at the model comparison section.

Long-Short-Term-Memory network

The input features of the LSTM network contain information on trace attributes and event attributes. All input features should have the same length; therefore, each trace should have the same number of events up to the decision moment. The trace length is chosen such that for most traces no events have to be removed and is set to 5. The updated event log is retrieved via Algorithm 6. The preprocessing as provided by Kratsch et al. [17] is used to retrieve the feature representation that can be used as input to a LSTM network. These feature representations will all have the same length and encode the trace attributes and event attributes up to the fifth event in the trace. It will also contain the dependent variable. The training and the validation data is used for hyper parameter optimization by a random search over possible hyper parameters. The best hyper parameters are used to train the final model. After training the LSTM model, the model is used with the test data set to generate predictions. For the test data set the model predicts whether the customer is expected to call VGZ. These predictions are compared with the ground truth of the traces respectively to determine the quality of the model. The orange bars in Figure 5.11 show the performance measures of the LSTM model. In the model comparison section the results will be discussed.

Generative Adversarial Network

The GAN model as proposed by Taymouri et al. [35] uses an encoder-decoder structure, to allow for variable length prefixes and suffixes. Therefore, the prefixes of the traces defined as the traces up to the decision moment can remain as the original prefixes. The suffixes of the traces defined as the events in the trace occurring on a later moment than the decision moment might require some adaptation. For the traces containing an event indicating that a customer has called VGZ in the suffix, the events up to the first call activity should be used. The remainder of the suffix should be removed as that part of the suffix is not of interest for the current research. Each row in the event log stores a single event describing the activity with the associated time stamp and case identifier. The activities should be mapped to integers, such that each activity is represented by an own integer value. The training, the validation and the test event log will be used as input to Algorithm 7. In this algorithm the training and the validation event log are used to train a GAN model with LSTM networks. The final model after training will be used to generate suffixes for the instances in the test set. The ground truth suffixes of the test set are available as well. The performance of the current model cannot be determined directly from the suffix. For each ground truth and generated suffix, it should be determined whether a call is occurring in the suffix. If no call activity takes place in the suffix, the dependent variable result of the suffix is 0 and otherwise it is 1. After extracting this information from the suffixes, the result is equal to the binary problem of predicting whether a call occurs on a later moment than the decision moment. The binary result is used to determine the performance measures for the GAN model. Figure 5.11 shows with gray bars the performance result of the GAN model. The results will be discussed at the model comparison section.

Random predictor

As already mentioned in Section 5.3, a random predictor is used as a baseline model to reason about the quality of the prediction models. To generate performance measures of the random predictor, the distribution of the traces containing the PoAc is needed. For the VGZ data set 3.48% of the traces contain a call on a later moment than that the customer has received feedback on the declaration as mentioned in Section 5.4.1. Creating predictions with the random predictor model is repeated 1000 times and for each iteration the accuracy, precision, recall and F1-score is calculated. The average over the 1000 runs is used to determine the performance measures of the random predictor. Figure 5.10 shows the result of the random predictor. In a random prediction task of a binary problem the probability of correctly classifying a data point is 50%. Therefore, the resulting accuracy and recall are expected to be equal to 0.5. Furthermore, the underlying data is imbalanced and the majority class is the class in which no call occurs. This class is defined as the negative class. For a random classifier this will result in providing more FP than FN predictions. Therefore, the resulting precision is expected to have a low score. Resulting in also a lower F1-score. These expectation are confirmed by the results of the random classifier shown in Figure 5.10.

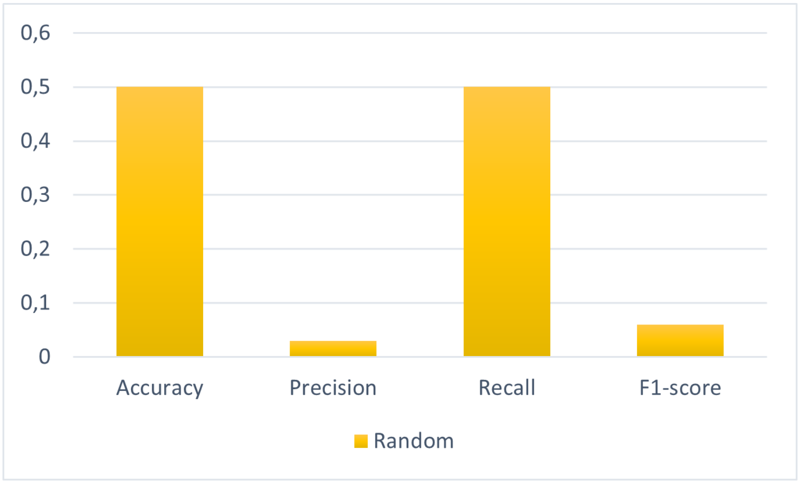


Figure 5.10: Performance measures of the random predictor for the VGZ data set

Model comparison

Figure 5.11 shows the performances measure of the RFC, LSTM, GAN and random predictor model. As can be observed, none of the models is performing best on all four performance measures. For VGZ it is mainly important to know whether a customer is likely to call. In consultation with domain experts it is decided that the recall together with F1-score are the two most important measures to consider. Considering these two measures the RFC and LSTM model are performing best. The GAN model is especially performing worse on the recall measure, while the F1-score for the random predictor is worse than the RFC and LSTM model. Considering the F1-score, the score of the RFC and LSTM model is doubled with respect to the random predictor. Therefore, the LSTM model and RFC are outperforming the random classifier. Considering all four performance measures, the RFC and LSTM model have equal performance; therefore, are expected to provide predictions with the same quality. Accordingly, considering the performance measures it cannot be decided which model is the best model to use. LSTM networks usually require a higher hardware requirement to train and use the model [17]. Furthermore, RFC

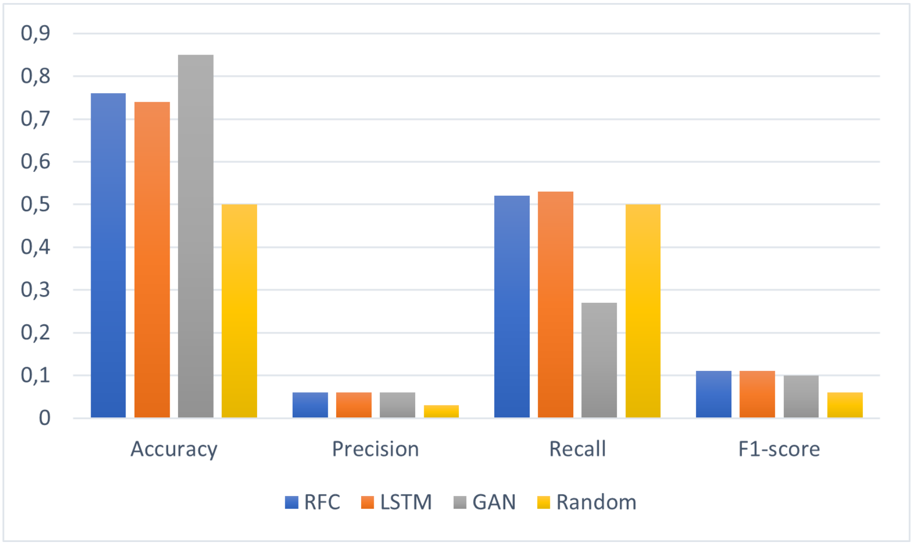


Figure 5.11: Performance measures of the prediction models on the VGZ data set

are easier to understand and explain for humans. Accordingly, the RFC might be selected as the best model for predicting whether a client will call VGZ after retrieving a result on the submitted declaration. Figure 5.11 shows a significant improvement for the RFC with respect to the random predictor. However, given the resulting values, the model still does not perform very well.

5.5.2 BPI 2012 challenge data set

This section shows how the different prediction models are applied on the preprocessed BPI 2012 challenge data set. Both decision moments, either defined as the first occurrence of 'A_PREACCEPTED' or the first occurrence of 'W_Completeren_aanvraag' are discussed. The data is preprocessed as discussed in section 5.4.2. The models will be trained to predict whether an offer will be sent to the customer in the remainder of its customer journey. Predictions should either be provided at the first occurrence of the event 'A_PREACCEPTED' or 'W_Completeren_aanvraag'. Next, the results of the random predictor for both decision moments are retrieved. Last, the different models are compared based on the performance measures.

Random Forest Classifier

To be able to train the RFC, traces should be transformed to features with the same representation as the example of Table 4.1. The prefix of the trace up to the decision moment should be used to create independent variables. Independent variables can be constructed from the events, event attributes and trace attributes available in the event log. The process of selecting independent variables is non trivial and no general applicable technique exists [8]. For each trace, the amount of loan or overdraft requested by the customer is known and can be converted to an independent variable. Furthermore, the number of activities, information on the types of activities and time between activities can be transformed to independent variables. The exact same independent variable should be extracted for each trace in the event log. The dependent variable can be extracted from list id_α and id_β returned by Algorithm 2. The customer whose case identifier is contained in id_α retrieves an offer and the dependent variable should be set to 1. The customer whose case identifier is contained in id_β does not retrieve an offer; therefore, the dependent variable is set to 0. The features of the log based on 'A_PREACCEPTED' will be different than the features of the log based on decision moment 'W_Completeren_aanvraag'. The activity 'W_Completeren_aanvraag' is executed on a later moment in the trace; therefore, the prefix will contain more events and the features will contain more information.

With the features created, a RFC can be trained. Training the model includes parameter tuning and selecting the best hyper parameters [29]. The best hyper parameters are determined using a randomized grid search. After the hyper parameters are tuned and the parameters of the RFC are trained, the model will be used to predict the binary class of the instance in the test set. The predictions are compared to the ground truth of the instances in the test set and used to determine the accuracy, precision, recall and F1-score of the model. The result of decision moment 'A_PREACCEPTED' are shown in Figure 5.14 and the result of 'W_Completeren_aanvraag' are shown in Figure 5.15. The performance measures of both decision moments are visualized with blue bars and will be discussed at the model comparison section.

Long-Short-Term-Memory network

In order to apply the feature creation as proposed by Kratch et al. [17] the number of events up to the decision moment should be equal for each trace. In the event log with decision moment 'A_PREACCEPTED', 81% of the traces contained in the event log contain 3 events up to the decision moment. Therefore, the trace length l as input for Algorithm 6 is set to 3. The updated event log returned by Algorithm 6 is used to create input features that are used as input to a LSTM model to predict the occurrence of the 'O_SENT' in the remainder of the trace. For the traces in the event log with decision moment 'W_Completeren_aanvraag', the trace length is chosen such that for the average trace no events have to be removed. On average, the prefix trace length up to the decision moment is 6 events and input l will be set to 6. Algorithm 6 will provide the updated event log that is used to create features of the traces that can be used to train a LSTM model to predict whether an offer will be sent to the customer. In the current data set, trace attributes are the amount requested and event attributes are the organisational resource and activity name. A random grid search over possible hyper parameters in combination with the training and the validation set is used to decide on the best hyper parameter values. The preferred combination of parameters are the hyper parameters used for final model training. Hyper parameter tuning will be conducted separate for both input logs. The test set is provided as input to the final model to provide predictions on the future occurrence of 'O_SENT'. The ground truth occurrence is compared to the predicted occurrence to provide insight into the model quality. The performance measures are shown with orange bars in Figure 5.14 and Figure 5.15 for decision moment 'A_PREACCEPTED' and 'W_Completeren_aanvraag' respectively. The results will be discussed at the model comparison section.

Generative Adversarial Network

The available training, validation and test log capture complete customer journeys. Only journeys up to the first offer sent to the customer on a later moment than the first occurrence of either 'A_PREACCEPTED' or 'W_Completeren_aanvraag' are required. Therefore, some of the traces available in the event log should be shortened, such that for each trace the trace is kept up to the first occurrence of 'O_SENT' on a later moment than the decision moment. For traces in which no event defining the activity 'O_SENT' is recorded on a later moment than the decision moment, the complete trace is contained in the log. Each event in the event log has an associated case ID, activity and a timestamp, which can be used to create the trace representation in the format of Table 4.3. The original activity names have to be transformed to a numerical representation. The feature representation is the input for the encoder-decoder GAN as proposed by Taymouri et al. [36]. First the GAN is trained using the traces available in the training and the validation set. Each trace is transformed to a prefix and suffix with respect to the decision moment. During training, the prefix as well as the suffix is available. When model training is completed, the prefixes of the traces in the test set are provided to the generator. The generator will provide suffix predictions for the instances in the test set. For each of these instances in the test set, the ground truth suffix is available as well. For the ground truth suffix as well as the predicted suffix it should be determined whether the activity 'O_SENT' is contained in the suffix, as this will provide information on the occurrence of the PoAc. The occurrences of the PoAc is used to determine the quality of the model. Performance measures of the GAN model on the BPI 2012 challenge log with decision moment 'A_PREACCEPTED' are shown in Figure 5.14 and with decision moment 'W_Completeren_aanvraag' in 5.15 by the grey bars. The results will be discussed in the model comparison section.

Random predictor

A random predictor, as introduced in Section 5.3, is used a baseline model to compare quality measures and to reason about the quality of the prediction models. As already mentioned in Section 5.4.2, for $67,2\%$ of the customer an offer has been sent to the customer in the event log with decision moment 'A_PREACCEPTED'. The fact that $67,2\%$ of the customer received an offer is needed to create the artificial data for the random predictor. In this case $67,2\%$ of the instances are assigned to contain the activity of interest. The final accuracy, precision, recall and F1-score performance score of the random predictor is the average over 1000 times repeating the random generation of the predictions. In the current research the prediction task is a binary prediction task. The probability of classifying a random guess correctly in a binary task is 50% for each guess. Therefore, the accuracy and recall is expected to be around 0.5. Currently, the positive outcome occurs in 67.2% of the cases; therefore, the predictions of the random classifier are more likely to be FN than FP. Consequently the precision is expected to be greater than 0.5. As the recall score is expected to be 0.5 and the precision to be greater than 0.5, the F1-score is also expected to be greater than 0.5. Figure 5.12 shows the performance measures of the random predictor. The figure indeed shows an accuracy and recall score around 0.5 and a higher precision and F1-score. Therefore, the performance of the random predictor is as expected.

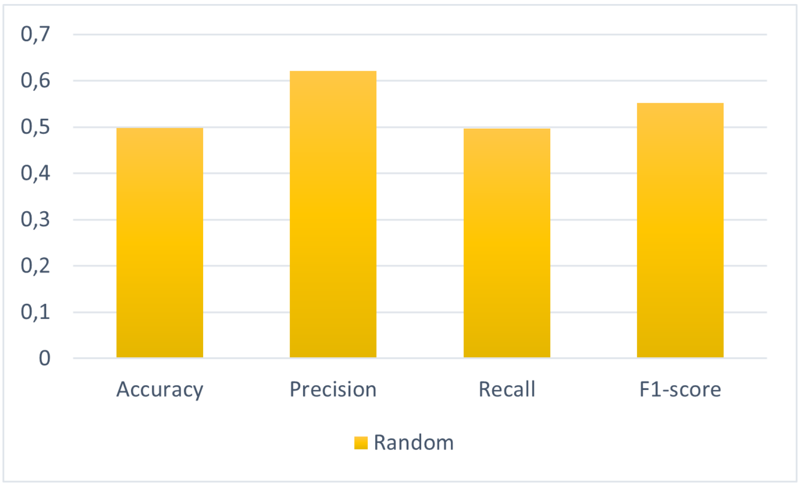


Figure 5.12: Performance measures of the random predictor for the BPI 2012 data set for decision moment 'A_PREACCEPTED'

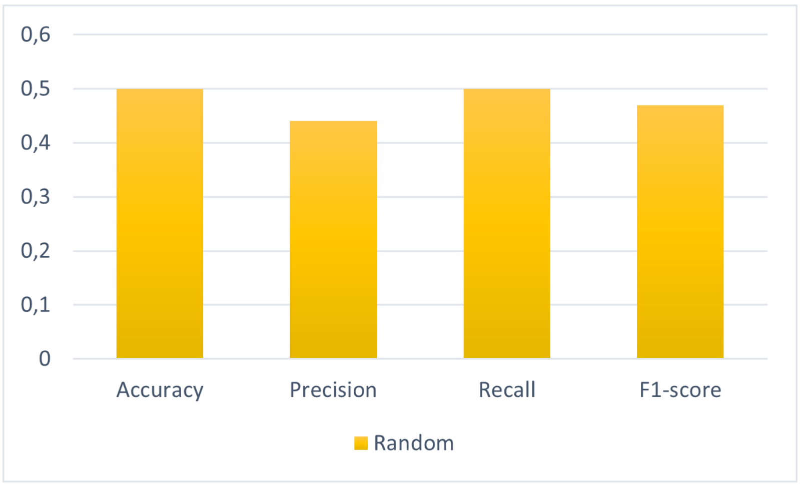


Figure 5.13: Performance measures of the random predictor for the BPI 2012 data set for decision moment 'W_Completeren_aanvraag'

A random predictor model should also be created as a baseline model for the event log in which the decision moment is defined as the first occurrence of 'W_Completeren_aanvraag'.

For this event log in 44.1% of the traces an offer is sent to the customer on a later moment than the first occurrence of ‘W_Completeren_aanvraag’ as mentioned in Section 5.4.2. The performance of the random predictor is determined by providing random predictions 1000 times. For each iteration the accuracy, precision, recall and F1-score performance score is determined. The final performance of the random predictor model is determined by taking the average of each score over the 1000 iterations. Predicting whether an offer will be sent is a binary prediction task. In a binary prediction the probability of a random guess to be correct is 50%. Therefore, the accuracy and recall is expected to be around 0.5. Currently, each prediction of the random classifier are more likely to be FP than FN, as the positive outcome occurs in 44.1% of the cases. Consequently the precision is expected to be smaller than 0.5. As the recall score is expected to be 0.5 and the precision to be smaller than 0.5, the F1-score is also expected to be smaller than 0.5. Figure 5.13 shows the performance measures of the random predictor. The figure indeed shows an accuracy and recall score around 0.5. The precision is indeed smaller than 0.5 and consequently the F1-score is also smaller than 0.5. Therefore, the performance of the random predictor is as expected.

Model comparison

Figure 5.14 shows the performance measures of the RFC, LSTM, GAN and random predictor for the BPI 2012 challenge log with the first occurrence of ‘A_PREACCEPTED’ defined as the decision moment. The random predictor is performing worst on all performance measures. Therefore, all three methods are outperforming the random predictor, especially on the recall and F1-score the machine and deep learning models have a higher

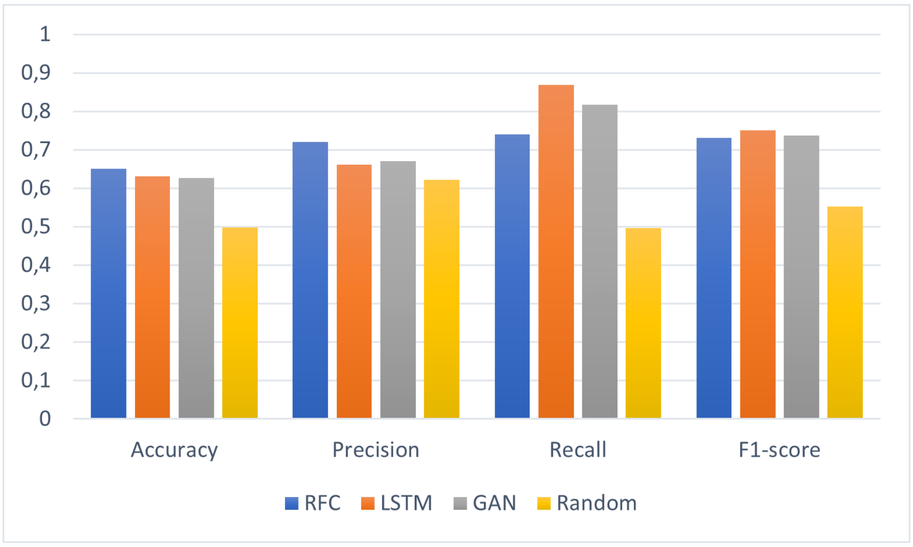


Figure 5.14: Performance measures of the prediction models on the BPI 2012 challenge log for decision moment ‘A_PREACCEPTED’

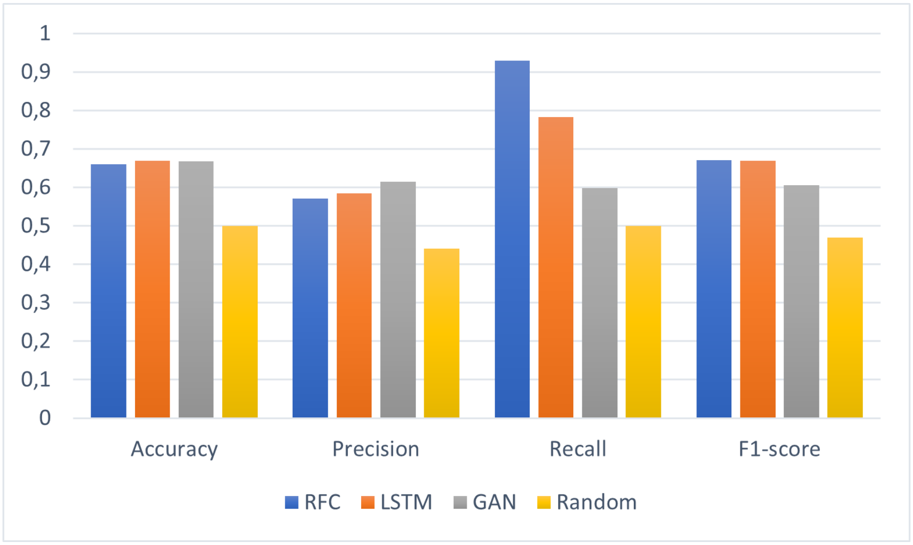


Figure 5.15: Performance measures of the prediction models on the BPI 2012 challenge log for decision moment 'W_Completeren_aanvraag'

measure. As can be seen in Figure 5.14, the accuracy and F1-score of the RFC, LSTM and GAN model are comparable. Furthermore, The LSTM model has the highest performance result on recall. Recall and F1-score are the two most important measures to consider during model comparison as discussed in Section 5.2. Therefore, in the current context the LSTM model is performing best to predict whether on a later moment in the trace an offer will be sent to the customer. The performance measure with respect to the random predictor have increased significantly.

The performance measures of the RFC, LSTM, GAN and random predictor for the BPI 2012 challenge log with the first occurrence of 'W_Completeren_aanvraag' defined as the decision moment are shown in Figure 5.15. The RFC, LSTM and GAN model are outperforming the random predictor on all four performance measures. The machine and deep learning models have comparable accuracy values. The RFC and LSTM model score equal on the F1-score and slightly better than the GAN model. The recall score of the RFC is outperforming the recall of the LSTM and GAN model. The LSTM model is still achieving a higher quality score on recall than the GAN model. Section 5.2 explained that the recall and F1-score are the two most important measures to consider during model comparison. As the RFC is scoring best on the recall measure, the RFC model is performing best in the current prediction task. The preferred model to provide future prediction to decide whether an offer will be sent is the RFC.

5.6 Results for training, validation and test set created randomly

This section describes the effect of creating the training, the validation and the test set time based or random. It demonstrates how the prediction results are affected by the creation of the training, the validation and the test set. In the research conducted up to this part the training, the validation and the test set have been created in a time based manner based on the times of the decision moment. If the different sets are created by randomly selecting instances, it might result in training on instances which are performed on a later moment than the instances on which the model is tested. In the real life scenario, it is not possible to train a model on future instances while predicting instances that are executed on this moment. Therefore, randomly creating a training, a validation and a test set may cause leakage of information and this may provide false insights in the quality of the model [32]. While, creating the different sets in a time based manner prevents the leakage of information. In this section the same research will be conducted as in Section 5.5, however, it will be conducted on a training, a validation and a test set that is created randomly instead of time based. The results will be compared to results discussed in Section 5.5. Section 5.6.1 discusses the result on the VGZ data set and Section 5.6.2 describes the results on the BPI 2012 challenge data set. For both data sets, almost all preprocessing steps for the data as well as the steps to create the models are conducted in the same way as in Section 5.5. Only the creation of the training validation and test set has been updated to be random instead of time based. On both data sets, for each model and each split of the data the results of training and testing the model once are discussed. Retraining a prediction model with the same data might produce slightly different results. For example, weights of the GAN model are randomly initialized. Different starting weights in the neural network can result in a different final model [13]. Therefore, the model can provide other predictions and performance measure can be different. For the random split of the training, the validation and the test data another split of the data will also result in different results and performance measures [40]. As in that case the model is trained and tested on a different training, validation and test set. Due to time constraints, only one random split has been used to retrieve the results in this section. However, it could be interesting in a future research to provide confidence intervals of the results and to check the conclusion of the current research.

5.6.1 Health insurer data set

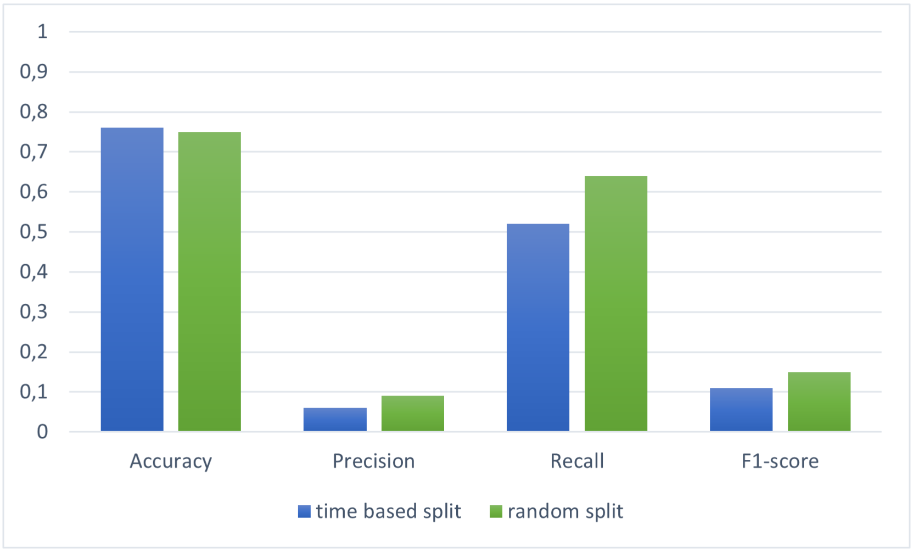
This section discusses the difference in performance of the RFC, LSTM and GAN model for the VGZ data set based on a time based split and random split to create the training, the validation and the test sets. Figure 5.16 shows the results for all three models. The blue, orange and grey bars show the performance of the model based on a time based split for the RFC, LSTM and GAN model respectively. The results of the random split are shown in green. Table 5.3 shows the training and inference time of the random split model. The training and inference times have not been recorded for the time based split,

Model	Training time in minutes	Inference time in seconds
RFC	106,36	2,20
LSTM	275,75	10,92
GAN	459,49	141,5

Table 5.3: Training and inference time of the random split data on the VGZ data

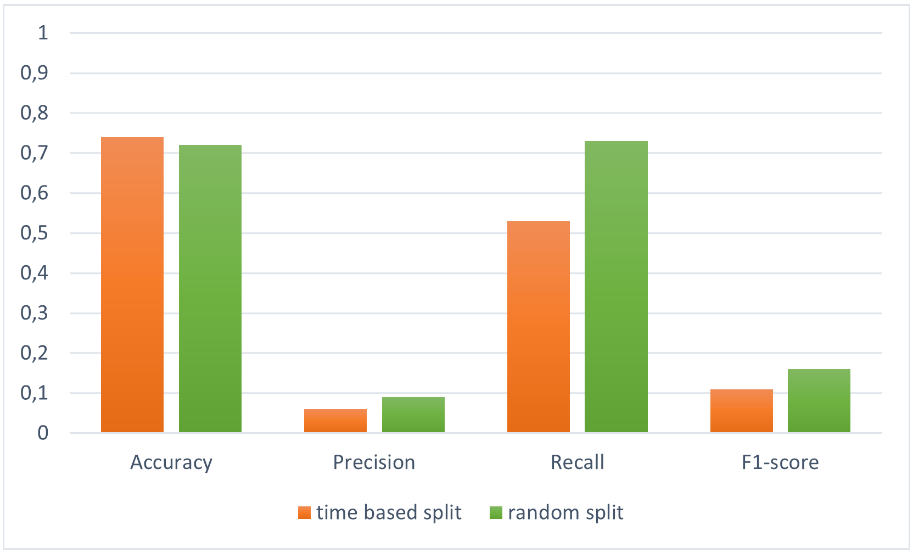
but are expected to be of the same order. It can be concluded that the running time of the GAN model is much longer than the time of the LSTM model. The LSTM model also needs a longer time to train the model and to infer results than the RFC model.

Figure 5.16a shows the result of the RFC model. The random split has a higher precision, recall and F1-score with respect to the time based split. However, the accuracy of the time based split is slightly higher, although it can be debated whether this difference is significant. Figure 5.16b shows the result of the LSTM model. Again, the random split has a higher score for precision, recall and F1-score. The accuracy score of the random split is again lower than the score of the time based split. The difference in accuracy is small; therefore, questionable whether the difference is significant. The result of the GAN model is shown in Figure 5.16c. The result for this model show a significant higher accuracy score for the time based split, while the recall is significantly higher for the random split. The performance measure of the precision is equal. For both splits the F1-score is slightly higher for the random split. All three models for the time based split are tested on the same test set, the models for the random split are also tested on the same

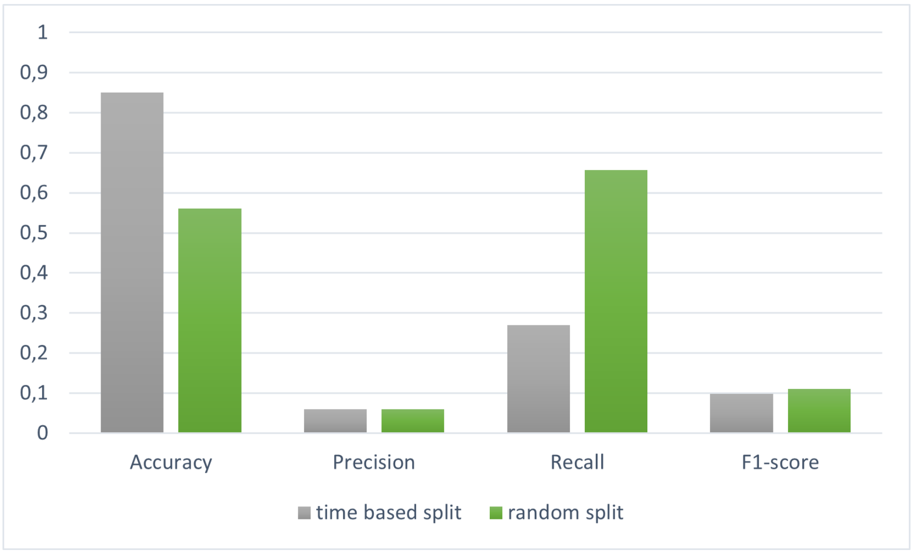


(a) Result RFC model

Figure 5.16: Performance measures of different training, validation and test set creations for the VGZ data set



(b) Result LSTM model



(c) Result GAN model

Figure 5.16: Performance measures of different training, validation and test set creations for the VGZ data set

set, however, the instance in the test set for the random split are not the same instances as the instance in the test set for the time based split. The difference of a higher precision, recall and F1-score which is not reflected to a higher accuracy can be explained by the distribution of a positive outcome in the test instances. Both test sets contain the same number of instances, however, the percentage of traces which contain a call in the test set has changed. For the time based test set 2.9% of the traces contain a call, on the other hand 3.6% of the traces in the random test set contain a call. Therefore, it is possible to retrieve higher results on the precision, recall and F1-score performance measure, while the general accuracy reduces.

As discussed in Section 5.2 the recall and F1-score are the most important measures to consider during model comparison. As the RFC, LSTM and GAN model on the random split score better on these measures, especially on the recall the model based on the random split would seem to be more promising in providing predictions on whether a customer will call VGZ. However, using a random split would provide performance measures which are most likely not representative for future predictions, as this may contain leakage of future information. The time based split is more representative for future unseen data and those result are more reliable. This indicates that using a random split might provide incorrect expectations for the quality of future predictions and should for that reason be avoided.

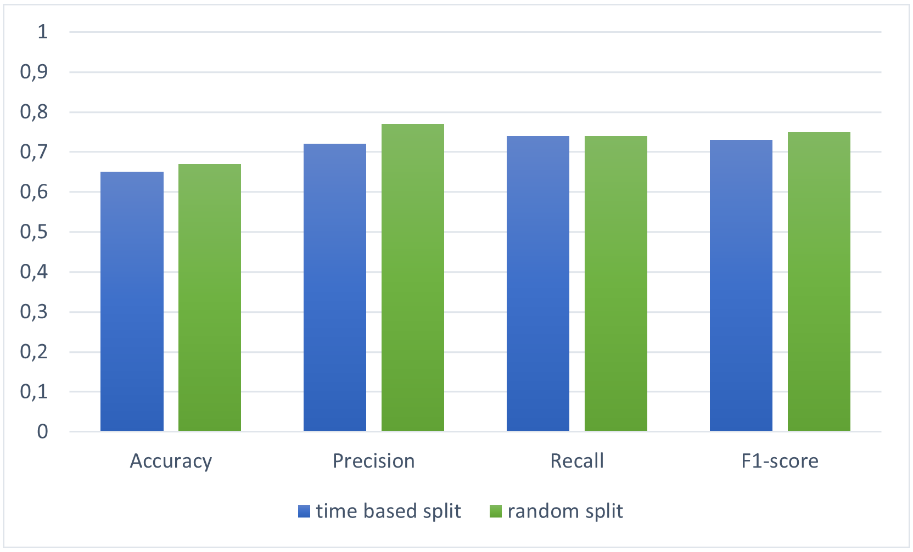
5.6.2 BPI 2012 challenge data set

This section discusses the different performance measures of the RFC, LSTM and GAN model on the BPI 2012 data set on the time based and randomly split training, validation and test data. The results will be discussed for the event log defined by decision moment 'A_PREACCEPTED' as well as 'W_Completeren_aanvraag'. Figure 5.17 shows the performance results of the three prediction models over the two generated training, validation and test sets of the BPI 2012 data set defined by decision moment 'A_PREACCEPTED'. The result of the RFC, LSTM and GAN model on the BPI 2012 data set with decision moment 'W_Completeren_aanvraag' is shown in Figure 5.18. For both figures, the results of the random split are shown in green. The blue, orange and grey bars show the performance of the model based on a time based split for the RFC, LSTM and GAN model respectively. Table 5.4 shows the training and inference times of both decision moments on all three models trained and tested on the randomly created training, validation and test set. The training and inference times for the time based split model are expected to be of the same order, but have not been recorded. For the models on both decision moments, the RFC model needs least time to train the model and infer results. The GAN model requires most time to train the model and infer results.

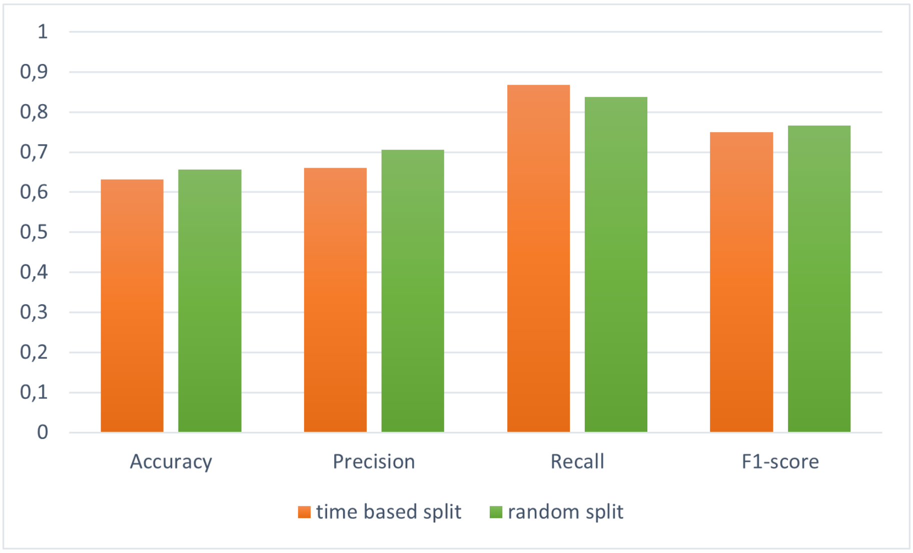
Decision moment	Model	Training time in minutes	Inference time in seconds
'A_PREACCEPTED'	RFC	43,20	0,45
'A_PREACCEPTED'	LSTM	100,40	0,54
'A_PREACCEPTED'	GAN	510,22	61
'W_Completeren_aanvraag	RFC	63,57	0,05
'W_Completeren_aanvraag	LSTM	157,58	1,21
'W_Completeren_aanvraag	GAN	1039,2	75,3

Table 5.4: Training and inference time of the random split data on the BPI 2012 challenge log

First, the result of the models trained and tested on the data set created with decision moment 'A_PREACCEPTED' are discussed. For all three prediction models the accuracy, precision and F1-score are higher for model trained, validated and tested on the randomly generated training, validation and test set. In contrast, the recall score for the LSTM model is lower for the predictor trained on the randomly split training, validations and test set as shown in Figure 5.17b. The recall measure is equal for both methods for

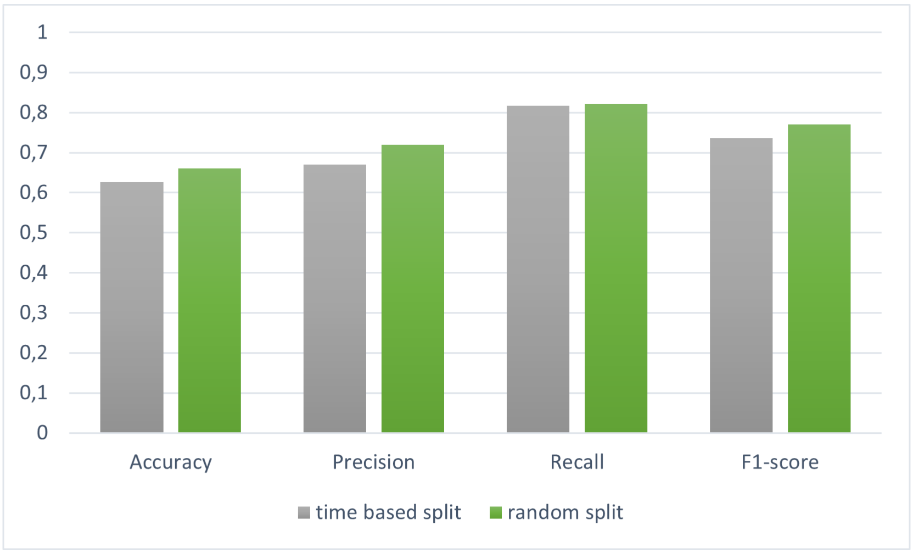


(a) Result RFC model



(b) Result LSTM model

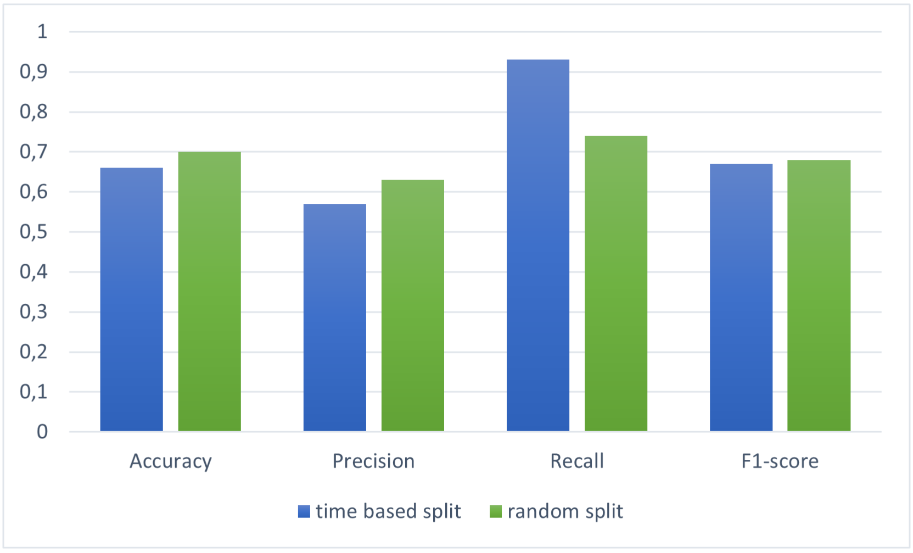
Figure 5.17: Performance measures of different training, validation and test set creations for the VGZ data set



(c) Result GAN model

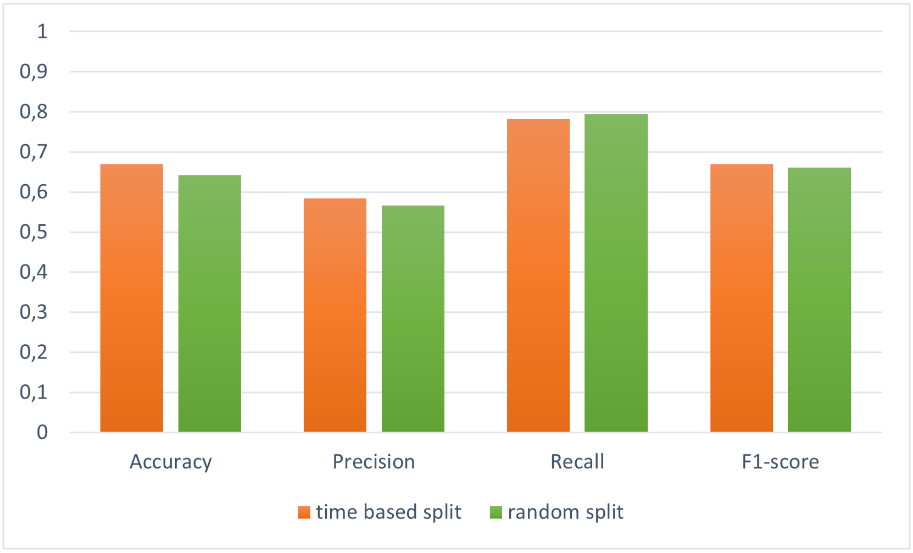
Figure 5.17: Performance measures of different training, validation and test set creations for the BPI12A data set with decision moment 'A_PREACCEPTED'

the RFC predictor shown in Figure 5.17a and GAN predictor shown in Figure 5.17c. In general it is not likely to retrieve a higher accuracy, precision and F1-score, while the

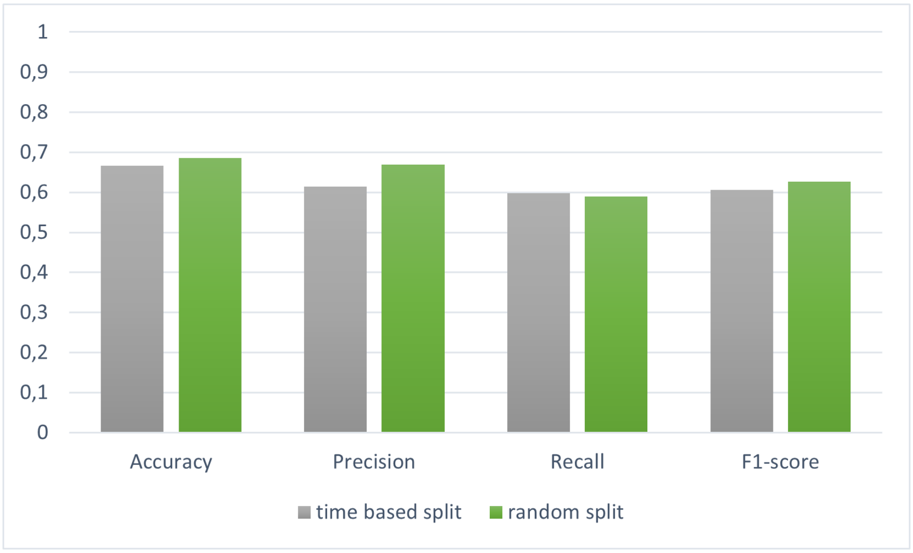


(a) Result RFC model

Figure 5.18: Performance measures of different training, validation and test set creations for the VGZ data set



(b) Result LSTM model



(c) Result GAN model

Figure 5.18: Performance measures of different training, validation and test set creations for the BPI12A data set with decision moment ‘W_Completeren_aanvraag’

recall is lower. However, it can be explained by the fact that the instances contained in the test set are different for both methods. In both cases the test set contains the same number of instances, but the instances are selected differently. Therefore, the number of positive and negative ground truth instances deviate and this results in a different sum of (TP+FN) predictions that will be predicted by the prediction model. The sum of (TP+FP) will be different as well. As the underlying distribution does not match it is possible to retrieve the current results. In the case of the RFC and the GAN models the models trained on the randomly selected training set are performing slightly better based on the performance measures. However, using a time based split is more reliable with future predictions as the time based split is better representing the real life scenario. Therefore, using a random split might provide unrealistic results for future predictions and it is better to use a training, a validation and a test set that is split in a time based manner. With the LSTM prediction model it is not possible to decide which model is performing better based on the performance measures. Due to the fact that the time based split has a higher recall, whereas the random split has a higher F1-score. In general the models are retrieving equal scores for all performance measures. Nevertheless, it is again better to use the time based split when training and test the model. The results of a such a model will always represent the real life scenario best and provide to most reliable insights.

Second, the prediction results for the three models on the event log defined by decision moment 'W_Completeren_aanvraag' are discussed. The results of the RFC model shown in 5.18a show that the accuracy and precision are higher for the model based on the random split. However, the recall for the model based on the time based split is much higher. The difference in F1-score in insignificant. Considering only the two most important performance measures, the time based split would be the preferred model to use. Figure 5.18b shows the performance measures of the LSTM model. The recall and F1-score difference in insignificant. The accuracy and precision of the time base split are slightly higher for the model trained and evaluated on the set created in a time based manner. For this model, again the model based on the time based split shows to be most promising. The results of the GAN model are shown in Figure 5.18c. For this model, the model trained on the time based split scores highest on none of the performance measures. However, the model trained and tested on the set which are randomly created shows some improvement on the accuracy, precision and F1-score. Comparing the two models on performance measures, the model based on the random split provides expectations on a more promising results for future predictions. However, using a random split might cause leakage of future information and therefore be unrepresentative for future predictions [32]. The models created with the time based split prevents leakage of future information while training the model. Therefore, a time based split should be used to generate performance results that are most trustworthy and most reliable to report while discussing the quality of a model.

5.7 Result of using a sliding window and landmark model

This section discusses the effect of using a sliding window or a landmark model for the creation of the training, the validation and the test set. Using a sliding window and landmark model provides insights in the trade off between resources needed to train a model and the prediction performance achieved by the model. A sliding window model only trains over the most recent instances, while a landmark model trains on the complete history of available data. Therefore, it is expected that a landmark model needs more resources to train a model. However, it is also expected that the quality of the predictions will be higher, as the model has more training data available. Section 5.7.1 discussed the result on the VGZ data set and Section 5.7.2 describes the results on the BPI 2012 challenge data set. For both data sets, almost all preprocessing steps for the data as well as the steps to create the models are conducted as in Section 5.5. Only the creation of the training validation and test set has been updated, such that sub windows of the complete data are used to create the data for the sliding window and landmark model. For this purpose, Algorithm 4 is used to create the different data set over the different windows. In the current research, all models are trained on a CPU. In the case that a GPU would be available the models could benefit from improved parallel computations [3]. Large models benefit more from a GPU as the architecture of larger models is better at exploiting the extra parallelism available at a GPU with respect to a CPU [42]. A GAN and LSTM network are expected to benefit more from a GPU, while the RFC is expected to be faster on a CPU. However, the result on the running time cannot be known in advance. Therefore, the conclusions provided later are based on the current results by only considering a CPU.

5.7.1 Health insurer data set

This section discusses the results for the sliding window and landmark model on the VGZ data set. The traces as filtered in Section 5.5.1 range over a time frame of 45 days ranging from May 18th until July 3rd. The data only contains complete traces. For a window ending at July 3rd, only complete traces will be included. For such a window, no trace exists in which the decision moment is contained in the time frame, while the trace has not been completed. Therefore, to ensure that each window has test instances, the last window considered should be finished just before July 3rd. For the sliding window a time frame of two weeks is considered and the window shifts with one week for each new window. For the landmark model, the first window does also contain the first two weeks of the data. The next windows each increase with the data of one extra week. Therefore, the third window contains four weeks of data. The results of the training time, inference time and performance measures are shown in Figure 5.19. The time indications at the horizontal axis in each graph show the end date of the window. Therefore, the start date of the window for the sliding window result is two weeks earlier. However, the start date of the window for the landmark model is May 18th for each end date. Each graph shows the results of the RFC, LSTM and GAN model on both the sliding window and landmark data. Results on the training time are shown in minutes and provides information on how long it takes to train the model on the training and the validation data. Inference time results are shown in seconds, providing the time it takes to provide predictions on the test set. The performance measures are provided with a separate graph for each measure.

As can be seen in Figure 5.19 the training time of the landmark model increases when the window size increases. The windows ending at June 15 or later, show that the prediction model trained on the data of landmark model needs much more time to train than the prediction model trained on the data available by the sliding window with the same end date. Furthermore, considering the same time window, the RFC is faster than

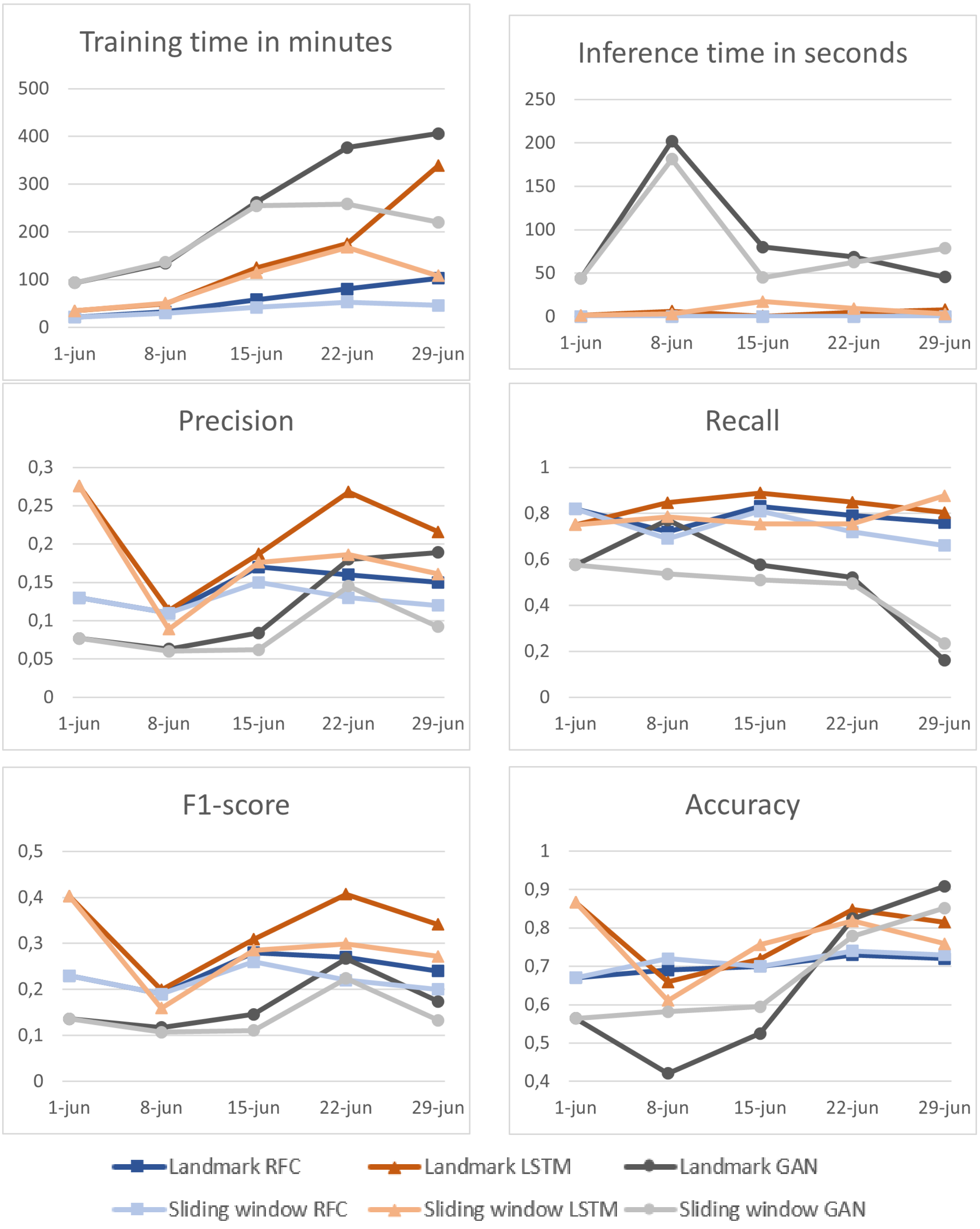


Figure 5.19: Training time, inference time and performance measures of the sliding window and landmark model for the VGZ data set

the LSTM model and the LSTM model is faster than the GAN model in training the model. Inference time for the RFC model and LSTM model is similar. For all windows the inference time is less than 30 seconds. However, the inference of the GAN model is slower.

Next to the training and inference time, the performance of the different models can also be compared. Performance measures of the three models over both windows are shown in Figure 5.19. For each prediction model considering the same end date of the time frame, the precision and F1-score of the landmark model is at least as high as the performance of the sliding window. For most windows, the landmark model is performing better than the sliding window. This shows that the landmark model is eager to learn using more journeys, even if these journeys are already a little older. Considering the recall score, up to the window ending at June 22, the landmark model is performing better than the sliding window for each prediction model. This changes for the GAN model and LSTM model trained on the window ending at June 29. For this window, the recall of the two models trained over the sliding window has a higher recall score than the two models trained over the landmark model. The accuracy of the RFC is equal for both the sliding window and the landmark model. For the other models it switches over time which model performs better on the accuracy performance measure. One can conclude that the GAN model is performing best on the accuracy measure for the window later in the time frame. While the F1-score and recall for the same model is worst at those windows. We can even notice that in a later frame the accuracy of the GAN model increases, while the recall drops. This shows that the GAN model is unable to identify the traces in which a call will occur as time progresses. On later time frames, the preference to predict no call to occur increases, as the accuracy increases and the recall decreases.

On all four performance measures, the LSTM model on the sliding window and landmark model scores higher than the RFC model as of the window ending on June 15. However, the training time of the RFC model is shorter. For a model to provide predictions, it is important to regularly update the model to new instances. Updating a model is easiest if training takes as less time as possible, but the results should not be affected by the reduction of the training time. The performance of the GAN model is not outperforming the RFC and LSTM model. The time to train a GAN prediction model on the data available by the landmark model is much longer and the recall and F1-score results of the final model are worse. The sliding window model also needs more time to train than any of the RFC and LSTM models and this model is performing worse on the recall, F1-score as well as the precision measure. Therefore, the GAN model is not the best prediction model in the current context. The performance on all four performance measures is slightly better for the LSTM model than the RFC model for the sliding window and landmark model. However, the training time of the LSTM model is also longer than the training time of the RFC model. Especially the time to train the LSTM model on the landmark model has already a long training time for the last window. Therefore, the landmark LSTM model is not the most preferred model. The gain in performance for the LSTM sliding window model is small with respect to the RFC model on the landmark model. The performance of the RFC model with the sliding window is again slightly lower. However, the model with the lowest performance requires the shortest training time. These three models could be considered to be the best model in the current context. As the RFC model is easier to understand, the RFC model is the preferred model to use. The running time of the landmark model of the RFC is not yet too long; therefore, the landmark model is preferred over the sliding window. However, if more data becomes available it is likely that the training time increases and that it will take to long to retrain a model using all historical data. In such a case, it is interesting to again conduct the sliding window research and investigate the effect of the size of the sliding window. As from that moment on, it might be interesting to only use the most recent data, but with a longer time frame. For example, six weeks of data could be used for the window.

5.7.2 BPI 2012 challenge data set

This section discusses the results for the sliding window and landmark model on the BPI 2012 challenge data set for both decision moments. The traces as filtered in Section 5.5.2 ranges from October 1st 2011 until March 14th 2012; therefore, the data contains records of 153 days. For a window ending at the last day in the time frame (March 14th), only complete traces will be included, as only complete traces are contained in the filtered event log. For a window ending at the last day of the time frame, no trace exists in which the decision moment is contained in the time frame, while the trace has not been completed. Therefore, to ensure that each window has test instances, the last window considered should be finished just before march 14th. In this research, a window size is chosen such that the complete frame is split in 5 windows. For the sliding window a time frame of 50 days is considered and the window shifts with 25 days for each new window. For the landmark model, the first window does also contain the first 50 days of the data. The next windows each increase with 25 extra days of data. Therefore, the third window contains 100 days of data. The results of the training time, inference time and performance measures for decision moment 'A_PREACCEPTED' are shown in Figure 5.20. Figure 5.21 shows the results of the event log defined with decision moment 'W_Completeren_aanvraag'. In both figures, the time indications at the horizontal axis in each graph show the end date of the window. Therefore, the start date of the window for the sliding window results is 50 days earlier. However, the start date of the window for the landmark model is the first of October 2011 for each end date. Each graph shows the sliding window and landmark model results of the RFC, LSTM and GAN model. Results on the training time are shown in minutes and provides information on how long it takes to train the model on the training and the validation data. Inference time results are shown in seconds, providing the time it takes to provide predictions on the test set. The performance measures are provided with a separate graph for each measure. First, the results of the application on the set with decision moment 'A_PREACCEPTED' are discussed. Second,the results of decision moment 'W_Completeren_aanvraag' are discussed.

Figure 5.20 shows the training time, inference time and performance measures of the application of the sliding window and landmark model on the BPI 2012 challenge log with decision moment 'A_PREACCEPTED'. The figure shows that the training time of the landmark model is longer than the training time of the sliding window of the same model

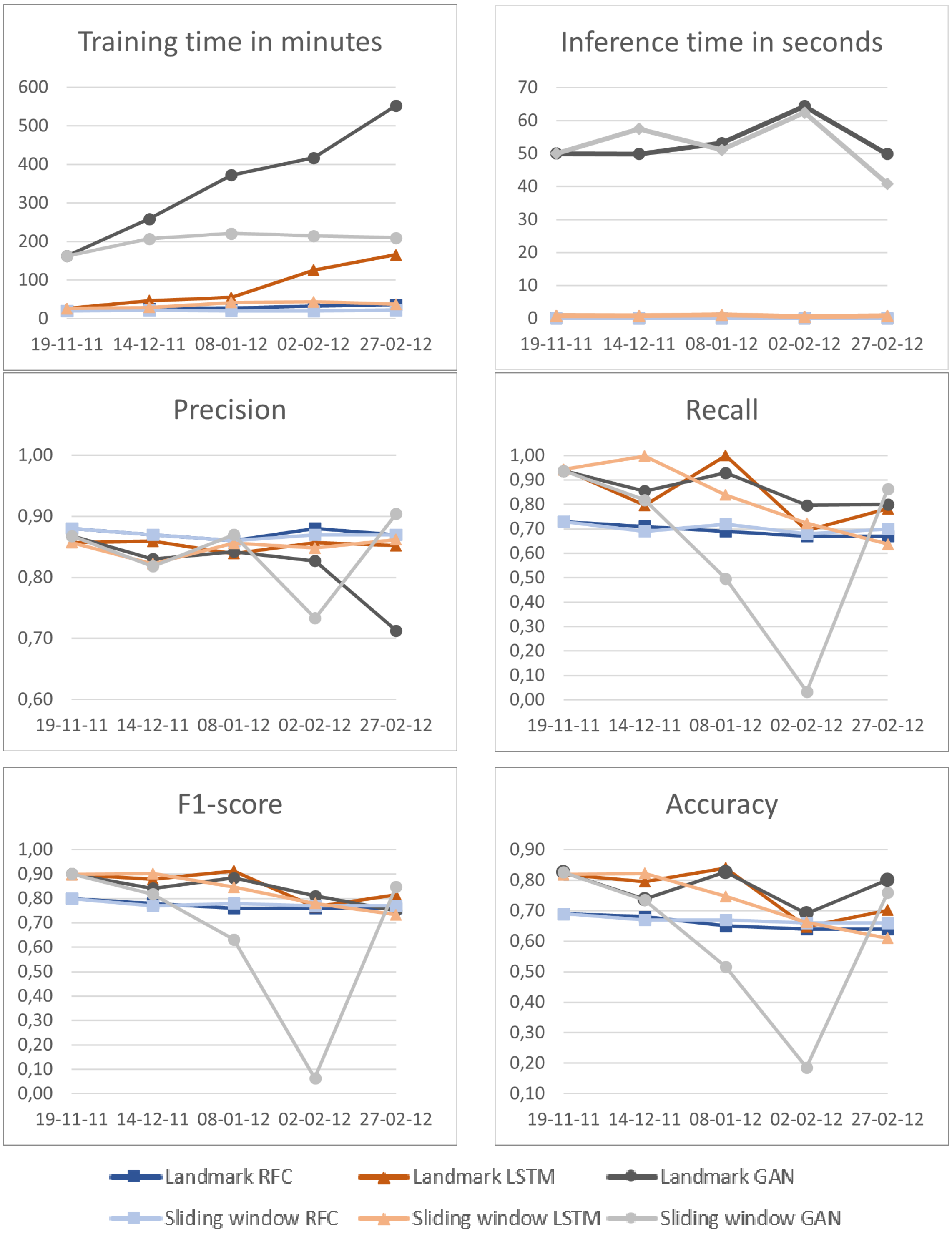


Figure 5.20: Training time, inference time and performance measures of the sliding window and landmark model for decision moment 'A_PREACCEPTED'

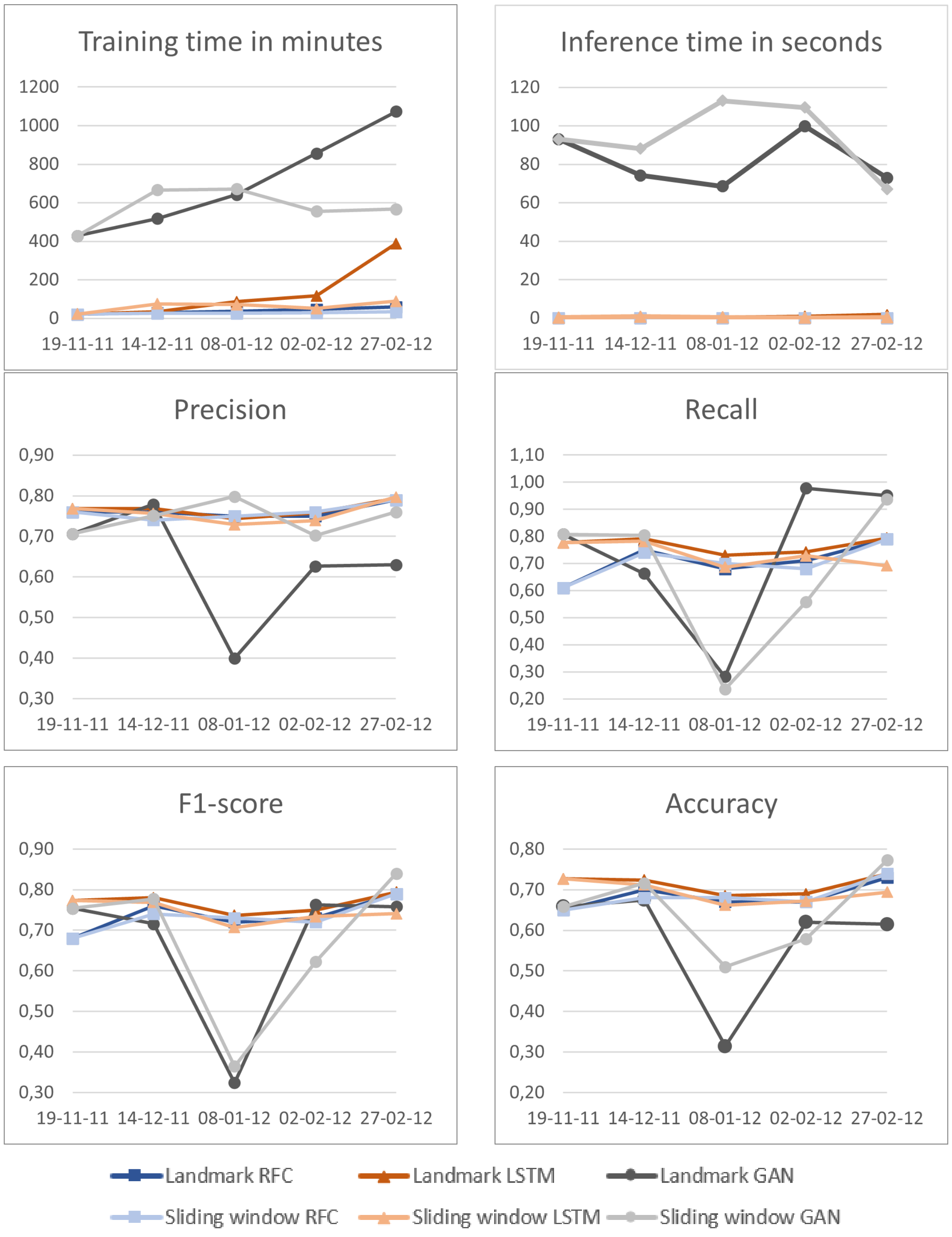


Figure 5.21: Training time, inference time and performance measures of the sliding window and landmark model for decision moment ‘W_Completeren_aanvraag’

for all three models. Furthermore, the training time of the sliding window is relatively consistent over the different windows, while the training time of the landmark model increases. The landmark model as well as the sliding window model of the GAN model have a higher training time than the other models for all windows. The training time of the sliding window model of the LSTM and RFC model and the landmark model of the RFC model have equal training times. All these models have a training time of less than 50 minutes. The last window of the landmark model for the GAN model has a training time of about 550 minutes and therefore needs 11 times more time train the model. Next to the slower training time of the GAN model, the inference also takes longer. The inference time of the sliding window and landmark model of the GAN model are comparable and take about one minute. The inference time of all other four models take at most three seconds. Therefore, these models are much faster in retrieving results on the test set.

Next to training and inference time, also performance measures are visualized. The GAN model with the sliding window is retrieving poor results for some windows. For the window ending at January 8th 2012 the recall, F1 and accuracy score drop. The window ending at February 2nd has even lower results. For this window the precision is also decreasing. Therefore, the GAN model has difficulties with predicting whether an offer will be sent to the customer for these windows. For the window ending at January 8th, it mainly has difficulties with predicting that an offer will be sent to the customer who based on the ground truth received an offer, as the recall performance is dropping. For this window, it is not yet predicting that an offer will be sent while no offer is sent too often, as the precision performance measure is not yet decreasing. However, for the window ending at February 2nd, it also predicts more often that an offer will be sent to a customer, while no offer is sent based on the ground truth value. A reason why the GAN model is not providing consistent predictions over the different time windows might be caused by the goal of the GAN model. The GAN model is trained to provide complete suffixes of the trace. Therefore, it is learning to provide prediction on the occurrence of multiple events and not only the event containing the PoAc. The GAN model trained on the landmark data provides good predictions for all windows. Only the precision is slightly dropping in the last window. Performance measure for the RFC model are consistent over all windows for the sliding window and landmark model. Both models have the same performance. The LSTM model is also consistent over the different windows. Again the models trained over the data available by the sliding window and landmark model are performing equal. Comparing the different prediction models, the GAN is worst as it has very poor results for some of the windows. The RFC and LSTM prediction model are both performing consistent, but the recall, F1-score and accuracy of the LSTM model is for most windows higher than the RFC model. The precision of the RFC model is slightly higher. As the recall and F1-score are the two most important measures to consider. The LSTM prediction model is performing best for both the sliding window and landmark models.

For models that need to be retrained often, it is important to consider the training time. Training time of the LSTM model with a sliding window or the RFC model with either the sliding window or landmark model needs the shortest training time. Therefore, considering the training time one of these three models is preferred. For all three models, the performance results of the landmark model is not increasing if more training data is available. Furthermore, for all windows the sliding window and landmark model have equal performance considering the RFC and LSTM model. Therefore, in the current context, no more data is needed to train a model that performs better. As a landmark model is in need of more training time if the window size increases, it is preferred to train a model using a sliding window. Predictions based on the LSTM model score highest on the recall and F1-score; therefore, it is best to use a LSTM prediction model with the sliding window for the current data.

Training time, inference time and performance results of applying the same research on the BPI 2012 challenge log with decision moment 'W_Completeren_aanvraag' are shown in Figure 5.21. It is again the case that the GAN model needs much more time to train than the RFC and LSTM model. The training times for the three prediction models trained with the sliding window are more or less consistent over the different windows for the same model. Training time for the landmark model increases as the window size increases and more data is available. The GAN model trained on the window ranging from October 1st 2011 until February 27th 2012 needs more than 17 hours to train. The RFC model of the sliding window and landmark model and LSTM model with the sliding window require much less time to train. For all windows the training time is less than 40 minutes. The inference time of the sliding window and landmark models for the RFC and LSTM model are equal. All these four models need less than 3 seconds for inference. However, the GAN model needs much more time to infer results on the test set. For all windows it needs between 65 and 120 seconds.

Next to comparing the training and inference time, the performance of the different models can also be compared. For the window ending at January 8th 2012, the GAN model has poor performance to predict whether an offer will be sent to the customer. Recall, F1-score and accuracy are all low for the GAN model, both for the landmark model and the sliding window model. Therefore, the two most important performance measures retrieve a low score for this model. The landmark model also has a low precision for this window. Training and inference of the GAN model includes providing predictions on the complete suffix of the trace. This suffix is used to decide whether the PoAc. However, predicting the whole suffix is more difficult than only predicting the occurrence of a certain event indicating that the PoAc occurred. Therefore, this might be the cause of the worse performance of the GAN model. On the other hand, the RFC and LSTM model do not show a drop in performance for the third window. These models are performing equal over the different windows. The models trained with the landmark model, contain more training instances at a later window. The performance of the RFC and LSTM prediction model trained over the landmark data do not show an increase in the performance. Therefore, the two models are not learning from more data and to retrain a model only the most recent data is needed instead of the complete historical data. The Recall and F1-score of the LSTM model trained over the sliding window data is slightly lower than the three other models. This difference is especially noticeable at the last window for the recall score. The results of the RFC model with respect to the sliding window and landmark model are really consistent. Therefore, it is impossible to indicate which of the two models is performing better. The difference in result with respect to the LSTM model trained on data available by the landmark model is also insignificant. Therefore, based on the performance measure the two RFC models and the LSTM model with the landmark window perform best and are equally good.

Combining the running times and performance result, it is possible to select the best model. Due to the drop in performance at the window ending at January 8th, the GAN model is not the preferred model to use in the current context. Based on the performance the landmark LSTM, landmark RFC and sliding window RFC have equal performance. The running time of the landmark LSTM model increases at later window and is larger than the running times of the two RFC models at the windows ending at February 2nd and 27th. Therefore, when the model should be kept up to date when new data becomes available, this LSTM model is less preferred. Up to the current window the running time of the RFC model trained on the landmark model is still almost equal to the running time of the sliding window. However, when more data becomes available for the landmark model, it is expected that the running time will increase. While the running time of the sliding window will be stable, as it will only consider the 50 most recent days. Since the performance of both models is similar, the RFC model with the sliding window is preferred.

5.8 Case study conclusion

In this section, the results of applying HIAP on both data sets will be interpreted. For both data sets the results of investigating the different applications will be discussed. For VGZ a recommendation will be provided on using the HIAP. These predictions indicate whether customers will call VGZ after receiving their result on a declaration.

The declaration customer log of VGZ contained a lot of different customer journeys. The resulting event log was skewed to the non-occurrence of the PoAc. Nevertheless, multiple approaches have been applied to predict whether a customer is expected to call VGZ after receiving a response on an earlier submitted declaration. First, RFC, LSTM and GAN models are evaluated on the complete data set. The models are applied on a training, a validation and a test set that are created by splitting the traces time based over the decision moments or randomly. For all three models the performance measures of the random based split seem more promising. However, randomly splitting the data provided a non-representative result, due to the possibility of leakage of information [32]. Therefore, randomly splitting the data should be prevented and splitting time based is preferred. The results of the time based split showed that the RFC and LSTM model resulted in the same performance quality. These two models provided better results than the GAN model. Next, the same data was used to perform research on the effect of using a sliding window or a landmark model for all three prediction models. The landmark model provided slightly higher performance measures than the sliding window model. But the training time to create the models is also higher. If time allows, it is useful to use the landmark model while training a model. If time is limited and the model should be retrained to remain up to date, a sliding window could suffice as well. The small reduce in performance can be worth the gain in time. Again, the RFC and LSTM models outperformed the GAN model. As the RFC model is easier to understand for humans, the RFC model is the preferred model to use.

Furthermore, the HIAP has also been applied on the BPI 2012 challenge data log. Two decision moments and one PoAc have been defined. The results to predict whether an offer will be sent to the customer with decision moment 'A_PREACCEPTED' are discusses followed by the results of 'W_Completeren_aanvraag'. Considering the data set with decision moment 'A_PREACCEPTED' and PoAc 'O_SENT' the RFC, LSTM and GAN model are used to provide predictions. For the implementation on the complete data set on which a training, a validation and a test set are created, the LSTM model was the best model to provide predictions. Considering the research in which the data has been split in windows to consider a sliding window and landmark model, again the LSTM model was the best model. In this case the sliding window and landmark model had the same performance. Since the sliding window takes less time to train a model, the sliding window LSTM model was preferred. The same research has been conducted on the data set with decision moment 'W_Completeren_aanvraag'. For the research in which the training, the validation and the test set had been created time wise over the moments of the decisions, the RFC and LSTM model had the highest performance on the F1 performance measure. Recall performance was the highest for the RFC model. Therefore, the RFC models was the best performing model. Investigating into applying a sliding window or landmark model resulted in three models having the same performance measures. In this case the RFC on the sliding window and the landmark model together with the LSTM predictor with the landmark model had the highest performance. As the training time for the sliding window in combination with the RFC was shortest, this is the preferred model. Therefore, the RFC model is performing best for both techniques when applied on the data set created with decision moment 'W_Completeren_aanvraag'. Additionally, the two different decision moments can be compared. The decision moment defined by 'A_PREACCEPTED' is earlier in the customer journey than 'W_Completeren_aanvraag'. For companies an early prediction is more interesting especially if the quality of the prediction is not lower for an early prediction. The performance measure of the prediction for the different decision moments are comparable. It is not the case that the performance of the decision moment of 'A_PREACCEPTED' is lower. Therefore, the predictions on whether a customer is expected to retrieve an offer can already be generated at decision moment 'A_PREACCEPTED'.

The result of both data set show that the main research question is answered. The future occurrence of an activity of interest can be predicted in a running customer journey by applying the HIAP. The subquestions mentioned in chapter 1 are also covered by the research. First the customer journey data is transformed to an event log and process models are discovered for the purpose of defining the critical moments. This shows a technique on how process mining techniques can be exploited in the context of customer journey analysis. Therefore, this contributes to the first subquestion. The second subquestion is addressed by comparing different prediction models. This shows that the result of the prediction is influenced by the prediction model. It even shows that different models can perform best depending on the context. The investigation of the effect of creating the training, the validation and the test set time based and randomly contributes to the third subquestion. As for that question the goal was to investigate whether the result of the prediction is affected by the creation of the training, the validation and the test set. It showed that creating the training, the validation and the test set randomly could provide quality results that most likely are to optimistic for the real scenario. The last subquestion is answered by applying the sliding window and landmark model. Again the prediction quality is affected by the windowing techniques. Next to the prediction quality it also provided insight in the difference in training and inference time of the different models. No windowing technique performed best in the entire research. Therefore, no general windowing technique can be advised to be used in general.

Chapter 6

Conclusion and future work

This chapter concludes this thesis. First, Section 6.1 summarizes the work of this thesis and discusses the conclusions. Section 6.2 discusses the future work.

6.1 Conclusion

In this thesis the focus was to fulfill two main objectives. First, bridging the gap between process mining and customer journey analysis, using them to improve the customer journey analysis. Second, defining a repeatable framework for future touchpoint prediction in a customer journey. To achieve these objectives HIAP is proposed. HIAP starts with a customer journey log. It predicts for customer journeys that are completed up to a defined decision moment whether the current customer is likely to interact with a specified touchpoint in the remainder of its journey. The customer journeys are converted to an event log. The potential activity might occur on any moment in the remainder in the trace, independent whether it is the next or last activity.

The first step in the framework is to convert a customer journey log to an event log. Steps that should be considered in this process are data cleaning and interaction transformation. Process mining techniques are applied on the retrieved event log to investigate the current process contained in the event log. These process models can be used to define the decision moment and potential activity. This approach bridges the gap between process mining and customer journey analysis contributing to the first objective. Furthermore, it exploits process mining techniques in the context of customer journey data. This contributes to the first research subquestion. It shows that customer journey analysis can be exploited with process mining techniques if the customer journey log is first transformed to an event log. The event log can be used as input to process mining algorithms to gain insights in the process recorded in the customer log. Once the decision moment(s) and potential activity are defined, separate training, validation and test sets can be defined to train prediction models. A RFC, LSTM and GAN prediction model have been exploited in this research. To define and compare the performance of the different models, performance measures are used. In addition to comparing the three models with one another, a random predictor is also defined as a baseline model. All these steps combined create a repeatable framework for the prediction of future touchpoints which contributes to the second objective. Furthermore, it also contributes to answering the second research subquestion. The research has shown that the three different prediction models achieved different performance results. Thus, the model influences the quality of the final predictions. The context of the data also provided a different model to be most suited. Therefore, one can conclude that no model can be selected as best model in general. Different models should be considered to decide which model is best in the context of the research.

HIAP has been applied to two data sets showing that the proposed pipeline is beneficial for companies dealing with customer journey data. The application has proven to provide potential activity predictions with a higher performance than the baseline model on the VGZ and BPI 2012 data set. The application on the VGZ declaration data shows decent improvements in the results, even though the results are not of the level that would be preferable for predictions. Additionally, the effect of creating the training, the validation and the test set time based over the decision moments or randomly has been investigated. This component contributed to answering the third research subquestion on how prediction results are affected by the creation of the training, the validation and the test data. For the different prediction models, most of the times the model created with the randomly selected instances for the training, the validation and the test set shows most promising results. However, randomly splitting the instances might create leakage of future information while training the model. Therefore, these more promising results might not represent the future model use on new instances correctly. Accordingly, creating the training, the validation and the test set should be performed on a time wise manner to create results that are most trustworthy. Furthermore, the effect of training models with a sliding window or landmark model is investigated. The models are compared quality wise by performance measures and time wise by the time it takes to train the model and to infer results. This part of the research contributed to answering the fourth research subquestion. As it examines the effect of applying various windowing techniques on model training and prediction quality. For all experiments conducted, the models trained over the data available by the landmark model had an equal or higher performance than the models trained with the data of the sliding window. However, it did not always show a performance gain, showing that not all models where able to provide better predictions if more data is available during training. The time to train a model always increased for the landmark model with respect to the sliding window. The sliding window training and inference time was more or less equal over all windows. The inference time for the landmark model was also comparable over the different windows. However, the training time increased over the later windows. As the gain in performance was small and in some cases insignificant for the landmark model, but the training time increased, the sliding window was preferred model option to use in some of the cases.

6.2 Future work

The research conducted leads to future interesting research. The HIAP has been applied to two data sets. It would be interesting to conduct the same kind of research on more data sets of different types of companies. This will provide insights in the fact whether the approach is indeed context independent and generic in the quality of the framework. In combination with applying the same research on multiple data sets, it is also interesting to investigate the effect of training models on the same data set multiple times. This might, due to some randomization in the models, create different results. This could be used to provide confidence intervals on the results obtained.

In the case study on the VGZ data set the results of the predictions show a low precision performance measure. A low precision indicates that it is predicted to often that a customer will call VGZ. In a future study, it would be interesting to provide certainty ranges together with the predictions. Meaning that it would not only be predicted that a customer will call, but that a prediction is provided that a customer will call with a $x\%$ certainty. This could provide insights in how likely it is that a customer will call and might result in distinct adaptions of the process for different customers.

During the case study predictions are provided about the occurrence of the activity of interest. As discussed it is preferable to retain activity predictions as early as possible in the process. Because early prediction provides more options to adapt the process to prevent the occurrence of the activity of interest. Next to early predictions, it might also be interesting to know at what moment the activity is expected to take place. Thus in a future study, next to predicting whether the activity of interest will occur, it could be worth investigating the possibility of predicting the time at which it would occur. This would provide information on the time that is left to adapt the process.

Bibliography

[1] W. M. P. van der Aalst. *Process mining: data science in action.* Springer Berlin Heidelberg, 2016.

[2] Stamatios-Aggelos N. Alexandropoulos, Sotiris B. Kotsiantis, and Michael N. Vrahatis. Data preprocessing in predictive data mining. *The Knowledge Engineering Review*, 34:e1, 2019.

[3] Shuichi Asano, Tsutomu Maruyama, and Yoshiki Yamaguchi. Performance comparison of fpga, gpu and cpu in image processing. pages 126–131, 08 2009.

[4] Eric Badouel, Luca Bernardinello, and Philippe Darondeau. *Petri Net Synthesis.* Springer-Verlag Berlin Heidelberg, 11 2015.

[5] James Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13:281–305, 03 2012.

[6] Gaël Bernard and Periklis Andritsos. Contextual and behavioral customer journey discovery using a genetic approach. In *Advances in Databases and Information Systems*, 09 2019.

[7] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[8] J Brownlee. *Machine Learning Mastery With Python.* Jason Brownlee, 2016.

[9] Marcos Cintra. *Genetic generation of fuzzy knowledge bases: new perspectives.* PhD thesis, USP- São Carlos, 04 2012.

[10] Boudewijn van Dongen. Bpi challenge 2012, 04 2012.

[11] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C, Prati Bartosz Krawczyk, and Francisco Herrera. *Learning from Imbalanced Data Sets.* Springer, 2018.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[13] Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture, 2018.

[14] Marwan Hassani and Stefan Habets. Predicting next touch point in a customer journey: A use case in telecommunication. *ECMS 2021 Proceedings*, 35, 2021.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[16] J.D. Kelleher. *Deep Learning.* The MIT Press, 1974.

[17] Wolfgang Kratsch, Jonas Manderscheid, Maximilian Roeglinger, and Johannes Seyfried. Machine learning in business process monitoring: A comparison of deep learning and classical approaches used for outcome prediction. *Business & Information Systems Engineering*, 63, 06 2021.

[18] K. N. Lemon and P. C. Verhoef. Understanding Customer Experience Throughout the Customer Journey. *Journal of Marketing*, 80(6):69–96, 2016.

[19] Benjamin Letham, Cynthia Rudin, and David Madigan. Sequential event prediction. *Machine Learning*, 93, 11 2013.

[20] Andreas Metzger, Adrian Neubauer, Philipp Bohn, and Klaus Pohl. *Proactive Process Adaptation Using Deep Learning Ensembles*, pages 547–562. Springer International Publishing, 05 2019.

[21] Douglas C. Montgomery and George C. Runger. *Applied Statistics and Probability for Engineers.* John Wiley & Sons inc., 2011.

[22] Igor Mozetiĉ, Luis Torgo, Vitor Cerqueira, and Jasmina Smailović. How to evaluate sentiment classifiers for twitter time-ordered data? *PLOS ONE*, 13(3):1–20, 03 2018.

[23] K. P. Murlphy. *Machine Learning, a Probabilistic Perspective.* The MIT Press, 2012.

[24] Anthony Myles, Robert Feudale, Yang Liu, Nathaniel Woody, and Steven Brown. An introduction to decision tree modeling. *Journal of Chemometrics*, 18:275 – 285, 06 2004.

[25] Suvi Nenonen, Sami Kärnä, Heidi Rasila, and Juha-Matti Junnonen. *Customer journey – a method to investigate user experiences.* CIB Publications, 01 2008.

[26] Dominic Neu, Johannes Lahann, and Peter Fettke. A systematic literature review on state-of-the-art deep learning methods for process prediction. *Artificial Intelligence Review*, pages 1–27, 03 2021.

[27] Márcia D. B. Oliveira, A. Guerreiro, and João Gama. Dynamic communities in evolving customer networks: an analysis using landmark and sliding windows. *Social Network Analysis and Mining*, 4:1–19, 2014.

[28] Avital Oliver, Augustus Odena, Colin Raffel, Ekin D. Cubuk, and Ian J. Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms, 2019.

[29] Philipp Probst, Marvin N. Wright, and Anne-Laure Boulesteix. Hyperparameters and tuning strategies for random forest. *WIREs Data Mining and Knowledge Discovery*, 9(3), Jan 2019.

[30] Brian Ripley. *Pattern Recognition And Neural Networks*, volume 11. Cambridge: Cambridge University Press, 01 2008.

[31] Cynthia Rudin, Benjamin Letham, Ansaf Salleb, Eugene Kogan, and David Madigan. Sequential event prediction with association rules. *Journal of Machine Learning Research - Proceedings Track*, 19:615–634, 12 2011.

[32] Johannes Ruf and Weiguan Wang. Information leakage in backtesting. *SSRN Electronic Journal*, 01 2021.

[33] Warren S. Sarle. Neural networks and statistical models, 1994.

[34] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with LSTM neural networks. In *Proceedings of the 29th International Conference on Advanced Information Systems Engineering*, pages 477–492. Springer, 2017.

[35] Farbod Taymouri and Marcello La Rosa. Encoder-decoder generative adversarial nets for suffix generation and remaining time prediction of business process models, 2020.

[36] Farbod Taymouri, Marcello La Rosa, Sarah Erfani, Zahra Dasht Bozorgi, and Ilya Verenich. Predictive business process monitoring via generative adversarial nets: The case of next event prediction. In *Business Process Management*, pages 237–256, Cham, 2020. Springer International Publishing.

[37] Alessandro Terragni and Marwan Hassani. Analyzing customer journey with process mining: From discovery to recommendations. *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 224–229, 2018.

[38] Alessandro Terragni and Marwan Hassani. Analyzing customer journey with process mining: From discovery to recommendations. In *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 224–229, 2018.

[39] Ikemefuna Udeh and Ngozi Eli-Chukwu. Artificial neural network based short term load forecasting in a power system network in nigeria. *International Journal of Trend in Research and Development 2394-9333*, 3:298–302, 08 2016.

[40] Muhammed Uçar, Majid Nour, Hatem Sindi, and Kemal Polat. The effect of training and testing process on machine learning in biomedical datasets. *Mathematical Problems in Engineering*, 2020:1–17, 05 2020.

[41] Gerrita van der Veen and Robert Ossenbruggen. Mapping out the customer's journey: Customer search strategy as a basis for channel management. *Journal of Marketing Channels*, 22:202–213, 09 2015.

[42] Yu Emma Wang, Gu-Yeon Wei, and David Brooks. Benchmarking tpu, gpu, and cpu platforms for deep learning, 2019.

List of Figures

List of Tables

Appendix A

Process model VGZ

This appendix contains the enlarged picture of the process model of the VGZ data set. In the main text, the image was not well-readable due to space limitations.

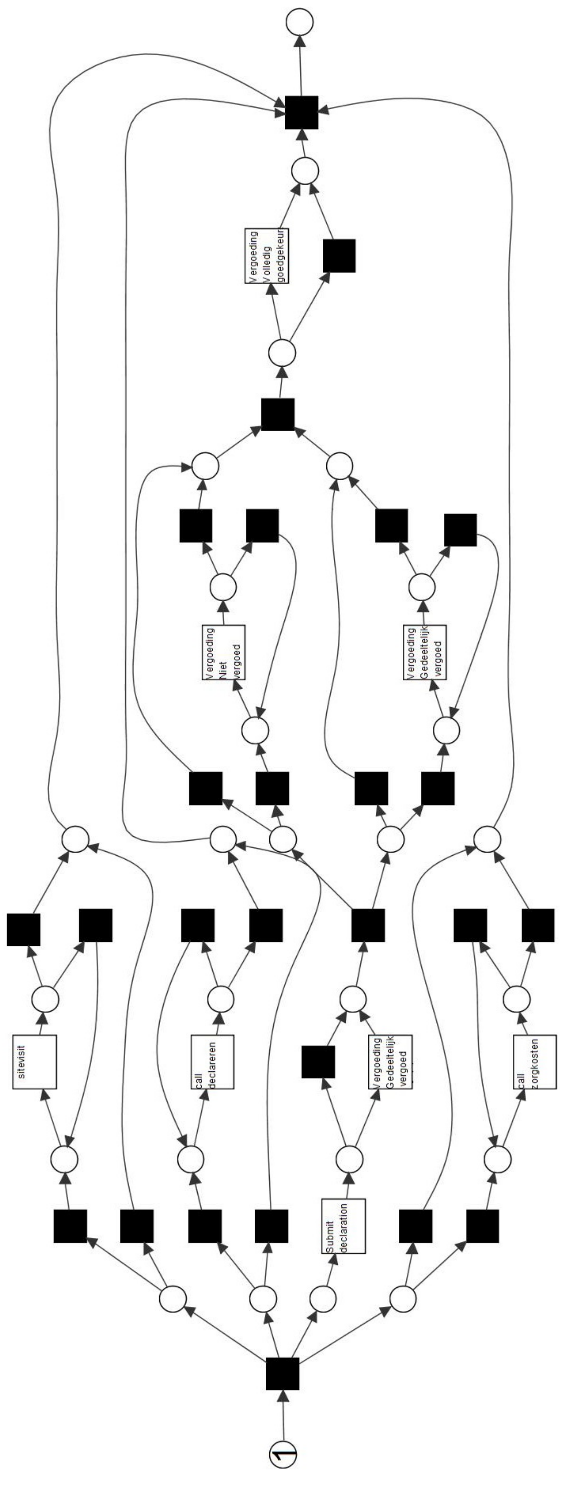


Figure A.1: Petri net of the trace in the VGZ data set with abstracted events