Predicting the occurrence of complaints within the customer journey based on process mining techniques

Nooyen, J.W.H.

Award date:
2020

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science
Architecture of Information Systems Research Group

Predicting the occurrence of complaints within the customer journey based on process mining techniques

Master thesis

JWH Nooyen

Supervisors:
Dr. Ing. Marwan Hassani
Marco Cordewener MSc.

Assessment committee member:
Dr. Ir. Irene Vanderfeesten

Eindhoven, January 2020

Abstract

Over the last couple of years customer journey analysis has received a lot of attention. Companies would like to know as much as possible about their customers and their behaviour to improve their organisation. Current research mainly focuses on the visualization of customer journey data, while data-driven techniques to analyze customer journeys are often missing. This thesis aims to fill one of the gaps by creating an approach that is able to predict the outcome of customer journeys that occur infrequently. This thesis introduces the Variant Based Outcome Prediction Approach (VBOPA) to provide, to the best of the writers knowledge, the first attempt to create a repeatable approach to predict the outcome of customer journeys. The approach combines elements from customer journey analysis, process mining and data mining by taking the best of each of the domains.

The approach is evaluated on a data set of a Dutch health insurer that contains customer journey data and where some journeys contain a complaint. The results show that the VBOPA reaches good prediction performance for the data set that shows the potential of the VBOPA. The VBOPA is also applied to the BPI 2017 Challenge log to show that it is generic and can also be applied to a different data set from a different context.

Keywords: customer journey analysis, process mining, outcome prediction

Preface

This master thesis is the result of the last step of my master program Business Information Systems at the Eindhoven University of Technology (TU/e). The research described in this thesis is the result of a collaboration between the Architecture for Information Systems group at the TU/e and two companies. The first company is a Dutch health insurer, the second company is a company that assists other companies in analyzing their customer journey data. I would like to thank both companies for giving me the opportunity to perform my graduation project under their supervision and for all help and feedback they have given me.

Furthermore I would like to thank Marwan Hassani for supervising the entire graduation project. Your guidance and feedback have been very helpful and is very much appreciated. I would also like to thank Irene Vanderfeesten for taking the time and effort to take place in the assessment committee.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the context of the thesis in Section 1.1. Then Section 1.2 describes the research problem and defines the research questions that are answered by the thesis. Finally Section 1.3 describes the structure of the document.

1.1 Thesis context

Nowadays companies are trying to learn how customers interact with a company and how these interactions are experienced. Lemon and Verhoef [20] state that customer experience currently is one of the most important considerations of management. Customers interact with companies via a variety of communication channels such as phone, e-mail, live chats and social media [22] in order to achieve a certain goal. Customers might for example interact with a retailer to purchase a product or with a travel agency to book a vacation. These interactions can be seen as a sequence that composes a customer journey. The analysis of customer journeys is proven to be useful for companies towards improving their organization [20]. A popular technique to analyze the customer journey is the creation of Customer Journey Maps (CJMs). A CJM is a visualization of all steps, often called touchpoints, at which a customer interacts with a company to achieve a certain goal, such as the purchase of a product [27]. CJMs are used by companies to gain an understanding of the main behaviour of customers and compare this with their expectations. If companies however come up with more complex questions that can not be answered based on CJMs, more complex techniques are required to provide an answer to such questions. This thesis aims to create an approach that is able to answer a specific type of question that is related to the decision making process of customers. Why does one customer decide to by a product while another customer does not? Or why do some customers decide to remain a customer while other customers do not?

Finding answers to such questions is very valuable for companies. Companies that know why their customers make certain decisions enables them to for example provide better product recommendations or take actions to prevent a customer from leaving the company, which might lead to a competitive advantage towards their competitors. This thesis has been performed in collaboration with a Dutch health insurer that has a similar goal. The Dutch health insurance market is a highly competitive market as Dutch residents can switch between different health insurers every year [37]. Therefore customer satisfaction is extremely important as satisfied customers are not likely to leave the company. Another reason why customer satisfaction is so important is that it is more costly for companies to attract new customers than to keep existing customer [25]. If the health insurer would be able to identify those customers that are less satisfied and are therefore thinking about churning to another health insurer, the health insurer can take specific actions to

prevent this from happening. An aspect that is quite related to customer churn are complaints as customers that have complaints tend to be less satisfied [3]. In order to identify the customers that might leave the health insurer, the health insurer would like to be able to predict which customers will file a complaint. If the health insurer would be able to identify those customers it can take actions to prevent the complaint or make sure the customer is taken care of in such a way that he or she remains loyal to the health insurer. The next section describes this problem in more detail and defines research questions.

1.2 Problem description

Concrete goals can be defined based on the thesis context. The solution that is proposed aims to provide an approach to predict which customers will file a complaint based on the customer journey they follow. The approach is called the Variant Based Outcome Prediction Approach, which is shortened to VBOPA, that aims to fill the previously identified gap. The VBOPA aims to make two main contributions.

The first contribution is that the bridge between process mining and customer journey analysis is further inspected. Previous studies have shown that process mining can be used to extract CJMs [2]. However, the proposed approach combines different elements of process mining with the world of customer journey analysis by inspecting different process models, often called process variants, instead of a single process model. These process variants are then combined with the customer journey data to enable the use of machine learning models.

The second contribution is that there is, to the best of the writers knowledge, currently no technique or framework available that performs predictions on customer journeys and fits the context as described in Section 1.1. The VBOPA therefore aims to provide a repeatable baseline approach for the given context that is able to perform predictions on customer journeys. The fact that this approach is repeatable makes sure that it can be applied to other contexts.

In summary, the main contributions of the thesis are:

- Bridging the gap between process mining and customer journey analysis by using the best of both worlds to improve customer journey analysis.
- Defining a prediction approach that is repeatable and can therefore be used as a baseline approach.

Based on the business problem and the proposed contributions the following main research question is defined:

How can the occurrence of complaints within the customer journey be predicted based on process mining techniques?

In order to make this research question more concrete, the following subquestions are defined:

- What is an appropriate measure to determine the similarity between two arbitrary customer journeys?
- What is the influence of the prediction model on the quality of the predictions?

1.3 Thesis outline

The remainder of the thesis is structured as follows: Chapter 2 describes related work in the context of this thesis. Chapter 3 contains notations that are required to understand the solution. Chapter 4 describes how the VBOPA is composed and explains all elements in detail. Chapter 5 describes how the VBOPA is applied to two data sets and evaluates shows how well the approach performs. Finally Chapter 6 contains the conclusions that are drawn and provides some future research directions.

Chapter 2

Related work

This chapter provides a description of the literature that is necessary to properly understand this thesis. Section 2.1 describes the concepts and core techniques of process mining. Section 2.2 describes the concepts of customer journeys and the link between customer journeys and process mining. Section 2.3 discusses the data mining algorithms that have been selected to predict whether complaints occur for customer journeys. This Chapter concludes with Section 2.4 where some related papers to the research problem.

2.1 Process mining concepts

Process mining is a research area that combines the domains of business process analysis and data mining. Process mining techniques aim to provide data-driven insights in processes [34]. Process mining techniques are applied to event logs. An event log is a collection of events that are executed at a certain point in time and are recorded by an information system. These events are executed for a specific process instance, which is known as a trace. A trace thus contains all executed events for a specific process instance that can be identified by a unique identifier [33]. Event logs are typically stored in the XES format, which is an XML-based structure and has been developed to provide a generic structure for event log data [13].

Figure 2.1 places event logs in the context of the "real world" on a very high level. Event logs can be extracted from these software systems, which contain recorded information from the "real world". This extracted event log can be used to apply process mining techniques to retrieve (process) models that accurately describe the "real world" such that analyses can be performed. There are three core types of process mining, highlighted in red, that can be applied are the following:

- **Process discovery:** Process discovery techniques aim to create a process model from an event log. In the ideal situation, this process model is simple and understandable, while also all instances from the event log fit the model [33]. An example of such a technique that is often used is the inductive miner [18].
- **Process conformance:** Process conformance techniques aim to inform the analyst how well an event log conforms to a process model. This type of techniques do require that a process model is already available. These techniques can be applied for multiple reasons: checking how well the actual data aligns to a normative model and inspect how well a discovered model actually fits the data [33].

- **Process enhancement:** Process enhancement techniques aim to improve a process model based on the event log. Again, these kind of techniques require a process model to be available. These techniques aim to extend a given process model based on aspects of the event log that are not present in the process model [33].

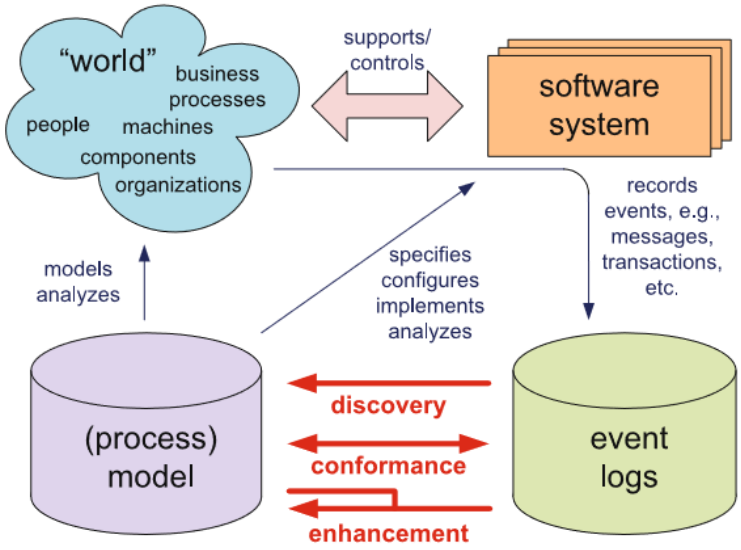


Figure 2.1: The three core approaches of process mining visualized, taken from [33]

The concepts of process mining are mostly used to define the scope and terminology within this thesis. Process mining is also used to gain an understanding of the data set in terms of the underlying process that leads to the occurrence of complaints. The tool that has been used to apply process mining techniques is the ProM framework [36].

2.2 Customer journeys

Følstad and Kvale have identified that there is no single clear definition of a customer journey, even though it is frequently observed within the literature. [8]. Meroni and Sangiorgi define a customer journey as "the repeated interactions between a service provider and the customer" [23], while Stickdorn et al. define a customer journey as "an engaging story about the user's interaction with a service" [30]. What is common between both definitions is that the perspective of the customer is central. Norton et al. [24] provide a more detailed definition that has been selected for this thesis:

> *A customer journey is the sequence of events, whether designed or not, that customers go through to learn about, purchase and interact with company offerings, including commodities, goods, services or experiences.*

Customer journey mapping is an approach that is, according to Følstad and Kvale [8], often applied to analyse customer journeys, usually by creating some kind of a visual representation. These visual representations are often called a customer journey map. Bernard and Andritsos [2] have identified the elements of customer journey maps based on a literature review, which are then placed in a Customer Journey Mapping (CJM) model based on the XES format [13]. Figure 2.2 shows the hierarchical structure of the CJM model as defined by Bernard and Andritsos.

A customer journey map is composed of multiple journeys, where a single journey is a possible path that a customer can take. A journey is followed by a customer, which interacts with the service provider. Usually some (demographic) information is known about each customer. A single customer journey consists of multiple touchpoints which can be ordered in time. A single touchpoint represents a single interaction between a customer and the service provider of which some properties are known such as the channel that has been used to perform the interaction.

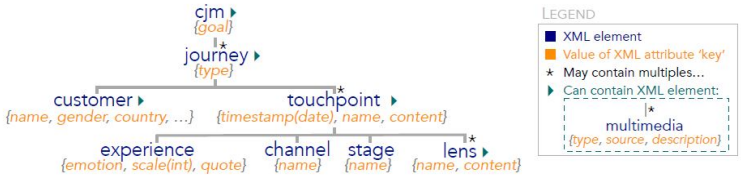


Figure 2.2: Hierarchical structure of the Customer Journey Mapping model as defined by Bernard and Andritsos [2]

The elements of the CJM model are then mapped to the XES format [13]. This is summarized in Table 2.1. The CJM model and this mapping enable a structured approach to apply existing process mining techniques on customer journey data and further research that is concerned with process mining on customer journeys.

CJM element	XES element
CJM	log
journey	trace
customer:name	trace:concept:name
touchpoint	event
touchpoint:name	event:concept:name
touchpoint:timestamp	event:timestamp:date

Table 2.1: Comparison between CJM model elements and XES elements

2.3 Classifier models

Classifier models are applied to data sets that are represented as a set of features and a single class label. The classifier models learn how to translate the set of features to the class labels. This type of machine learning problem is also commonly referred to as a supervised learning problem as the class labels are known [16]. Supervised learning techniques are, as also stated by Kotsiantis et al., applied to create a model that learns about relations between the features that is able to accurately classify the class labels [17]. This is conceptually visualized in Figure 2.3.

There are several types of classifier models, of which three are selected: logistic regression, random forest and neural networks. These three models have been selected for a number of reasons. Firstly, these types of models have been applied successfully in a number of studies and in different domains. Secondly, these three models represent three different types of models in terms of complexity and consequently interpretability. Logistic regression models are mainly based on statistics and are therefore relatively simple. This means that they can be interpreted quite easily, although they might not reach a desired level of performance for complex data sets. Neural networks are most difficult to interpret of these three models, but it has been proven that it is able to learn complex relations between features that can lead to superior prediction performance. Random forest models are in between logistic regression and neural networks in terms of complexity and interpretability. Random forest models are ensemble predictors as they

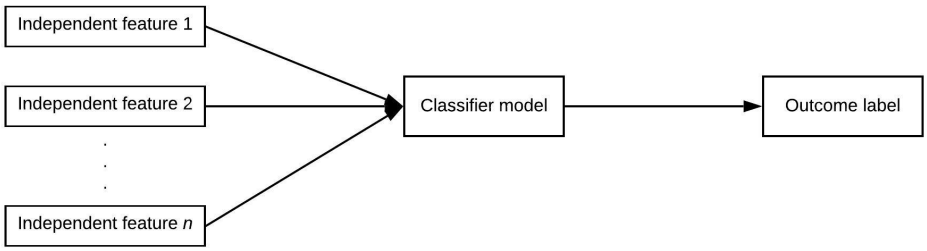


Figure 2.3: Conceptual overview of a classifier model

are constructed from a number of decision trees. This limits the interpretability compared to logistic regression but is simpler than neural networks. It has been proven that they are able to reach a good prediction performance for very large data sets. By selecting these three models a good distribution in terms of complexity and interpretability is ensured.

Logistic regression model

Logistic regression is a generalization of linear regression and is often used to predict a binary class variable. Schein and Ungar define a logistic regression model as a model that calculates the odds of a certain binary outcome based on a set of predictors (i.e. independent features) [28]. A logistic regression model, conceptually represented by Formula 2.1, is composed of an intercept and a coefficient for each independent feature. The intercept is represented by β_0 and the independent features and corresponding coefficients are represented by x_i and β_i respectively. The learned model classifies an instance by replacing all x_i by the values for the corresponding features of the instance, which results in a certain probability that is used to determine the class label. The model is thus relatively simple and can therefore be easily interpreted by analyzing the size and sign of the coefficients, as this can be used to reason about the impact of a high or low value for a certain feature on the probability.

$$Probability = \beta_0 + x_1 * \beta_1 + x_2 * \beta_2 + ... + x_n * \beta_n \tag{2.1}$$

Random forest model

Ensemble models are a collection of predictor models that are trained for the same purpose. Each predictor model is trained by using a different subset of the data, which are aggregated into the final ensemble model that is used to perform predictions. Ensemble models are used to reduce variance and improve the prediction performance [29]. Random forest models are a collection of decision trees, hence it is an ensemble model, and has been defined by Breiman [5]. Breiman proved several strengths of the random forest classifier: it is efficient for large data sets and a large number of independent features and it is robust against noise and outliers. A random forest model is very simple and interpretable, but its components (i.e. the decision trees) are very easy to interpret and analyze. Following Rokach and Maimon, a decision tree classifier is defined as a rooted tree that consists of a number of nodes [26]. Each instance starts at the top node, which is also called the root node, where the first decision is made. Then the tree is traversed downwards via a number of internal nodes and eventually ends up at a certain terminal node where the class label is determined. At each internal node the path is decided based on the value of a certain feature. An example of a decision tree is shown in Figure 2.4, where internal nodes are represented

by circles and terminal nodes by triangles. At the root node a split is made based on the age. If the age is greater than 30, a terminal node is reached and the class label is *No*. If the age is equal to or less than 30, an internal node is reached where the path is decided based on the gender. A random forest model essentially does the same thing to classify instances, but at each internal node the decision is based on the ensemble of decision trees instead of a single decision tree.

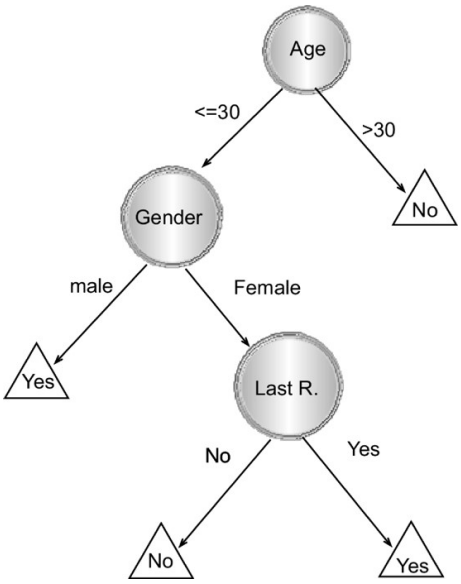


Figure 2.4: Example of a decision tree classifier for two possible classes: Yes and No. Figure taken from [26]

Neural network

The idea of neural network models is based on the human brain and the way the human brain processes information. A neural network is composed of a number of neurons, similar to the human brain, that receive input information which is processed by the neuron after which a certain output value is provided. The way the neuron processes the input information depends on a combination of weights and an activation function. [14], which is already quite difficult to understand. A neural network is composed of a lot of neurons that are placed in layers and all neurons between each layer are often fully connected. A visual overview of the structure can be found in Figure 2.5. The way a single neuron processes the input data is already quite difficult to understand, let alone to understand how an entire neural network reaches its prediction. This type of classifier model is therefore very difficult to understand and can be considered as a black box model: it is only known what data is provided as input and what is returned as output, but it is not known what happens inside the black box.

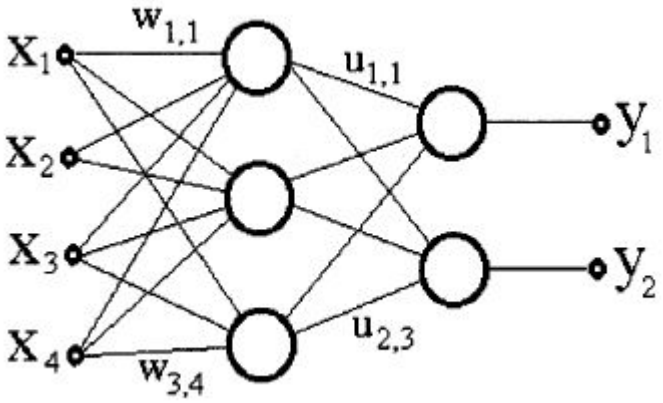


Figure 2.5: Example of a neural network model. Figure taken from [14]

2.4 Related studies

This section describes several studies that are related to the research problem. There are two topics of studies that are discussed. The first topic of studies are related to distance measures that are used to compare traces to each other or to process models. The second topic of studies are related to predictions that are performed on customer journey data.

2.4.1 Distance measures

Tax. et al [31] have defined an approach that is concerned with the quality of event log data. They identify that event logs often contain events at a very detailed level which has as a consequence that process discovery algorithms are not able to mine understandable process models. They define an approach that uses supervised learning techniques to abstract the detailed events to a higher level of abstraction. The approach is evaluated by comparing the abstracted traces to each other and calculating the similarity between the traces. Such a similarity measure might be useful to compare customer journeys as well. Tax. et al use similarity calculation that is based on the Levenshtein distance [21], which is shown by Formula 2.2.

$$Levenshtein\_similarity(a,b) = 1 - \frac{Levenshtein\_distance(a,b)}{max(|a|,|b|)} \tag{2.2}$$

The Levenshtein distance [21] was originally defined to compare the similarity between two string values. The Levenshtein distance calculates the minimum number of insertion, deletion and substitution operations that need to be performed on one string to transform it into the other string and each operation has a cost of 1. In essence, strings and traces are very similar to each other as a string is a sequence of characters while a trace is a sequence of events and therefore the Levenshtein distance can be easily applied to event log data. This was also observed by Tax. et al and they used the Levenshtein distance as is shown in Formula 2.2. The formula also has the nice property that the similarity values will always be between 0 and 1 because the Levenshtein distance between two traces is divided by the length of the longest trace. This similarity calculation seems to be suitable for the research problem of this thesis.

Cordewener [7] has recently performed a study at a Dutch health insurer to develop a technique to identify customer journeys from an event log. This study is obviously interesting for the research problem of this thesis as the research is done in collaboration with a Dutch health insurer as well.

Cordewener created a technique that is able to divide a trace containing all contact registrations of a single customer to a number of customer journeys. The idea is that a customer might interact with the health insurer for multiple reasons, or in other words to reach different goals, at the same time which has as a consequence that the trace containing all contact registrations does not accurately represent the customer journey. The technique transforms the events of a trace to a profile vector that is used to run a clustering algorithm that clusters the events of each customer journey.

Even though this technique does not seem applicable for the research problem of this thesis, some elements of the work of Cordewener might be useful. Cordewener defines an approach to calculate how similar events are to each other based on a variant of the market basket analysis. It is evaluated for each pair of events A and B how often event A is directly followed by event B in the event log, which is then divided by the amount of times event A occurs in total in the event log. Such an approach might prove to be useful when defining a metric to compare arbitrary customer journeys.

2.4.2 Predictions on customer journeys

Goossens [10] has defined an approach called OARA, which stands for Order-Aware Recommendation Approach. The approach is able to perform predictions and recommendations on customer journey data. The kind of predictions that are performed are predicting the next event that will occur for a customer journey. These predictions are performed by including features based on the previous events of a customer journey and using machine learning to retrieve a model that performs predictions. Furthermore, the predictions are used to perform recommendations as well.

The technique is not applicable for the research problem of this thesis as the goal of the approach is to provide recommendations. Also the context of the study is a company that sells lighting products where it can expected that the type of customer journeys are very different than the customer journeys of a health insurer. However, the idea behind the approach can serve as a starting point for finding a solution to the research problem of this thesis. The way Goossens combines process mining and machine learning shows promising results and a similar idea might definitely turn out to be a good solution.

Terragni [32] also defines an approach to perform recommendations based on customer journey data in a web environment. The approach of Terragni uses process mining to identify customer journey paths that are optimal in terms of a certain KPI value. Then recommender system algorithms are used to try and force people to follow a customer journey path that reaches the highest KPI value.

Again, this is a different context and it is expected that the type of customer journey data is very different from the customer journey data of the health insurer. The idea of identifying those customer journey paths that maximize a certain KPI value could be adapted towards the research problem of this thesis. Instead of identifying the customer journey paths that maximize a KPI the customer journey paths where a complaint occurs can be identified. These paths could then be analyzed in a certain way to determine why certain customers follow this customer journey path and why other customers do not.

Chapter 3

Preliminaries

This chapter describes preliminaries that are required in order to understand the framework that is described in Chapter 4. Section 3.1 describes all notations related to the data structures and Section 3.2 describes notations related to the comparison of traces.

3.1 Notations

This section describes the notations that are used in subsequent chapters. Let

$$CI = (ci_1, ci_2, ..., ci_n) \tag{3.1}$$

be a customer interaction log that is extracted from a database that registers customer interactions. Within this customer interaction log, each element

$$ci_i = (customer\_id, t, s) \tag{3.2}$$

defines that the customer that is identified by the *customer_id* has had an interaction at time t about subject s. The *customer_id* is used to uniquely identify a customer. From the customer interaction log two sets can be extracted: the set of customers $CU = \{cu_1, cu_2, ..., cu_{|CU|}\}$ and the set of subjects $S = \{s_1, s_2, ..., s_{|S|}\}$.

Furthermore, an event log L is defined as

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

where each element is an event, which is defined as

$$e_i = (c, \tau, a) \tag{3.4}$$

Each event consists of a case identifier c that uniquely identify the case to which an event belongs, the timestamp τ that indicates when the event occurs and the activity label a that describes what event has occurred. Again, two sets can be defined: the set $C = \{c_1, c_2, ..., c_{|C|}\}$ that contains all case identifiers and $A = \{a_1, a_2, ..., a_{|A|}\}$ containing all activity labels. Finally,

a trace $tr =< a_g, a_h, ..., a_n >$ is the ordered sequence of the activity labels of the events that are observed for a single case.

The customer interaction log CI and the event log L show a very similar structure and therefore a mapping between the two is quite straightforward. Each *customer_id* can be mapped to a case identifier c, each subject s can be mapped to an activity label a and the timestamps t and τ can also be mapped to each other easily.

3.2 Trace comparison

This section describes several notations that are used to compare two arbitrary traces based on the event log L. An Event Similarity Matrix (ESM) is defined to accommodate the comparison of arbitrary traces. An ESM contains the elements of set A on both the column and row indices and is defined as follows:

$$\begin{aligned} ESM = ((&sim_{a,a}, sim_{a,b}, ..., sim_{a,|A|}),\\ &(sim_{b,a}, sim_{b,b}, ..., sim_{b,|A|}),\\ &.....,\\ &(sim_{|A|,a}, sim_{|A|,b}, ..., sim_{|A|,|A|})) \end{aligned} \tag{3.5}$$

Each element $sim_{a,b}$ corresponds to the similarity between the activity labels a and b. The values of the cells of the ESM can be determined based on domain knowledge or based on the data of event log L. The most optimal approach depends on the situation at hand. Domain experts might have knowledge that can not be extracted from the event log L and might thus be preferred. When the number of activity labels becomes too high it might be too time consuming for domain experts to create the ESM by hand and an approach based on the event log L might be preferred. An approach based on L is described in detail in Chapter 4. Independent of the chosen approach, the ESM should adhere to the following properties:

- Each possible pair of activity labels should be present in the ESM. Consequently, the number of columns and rows of the ESM should be equal.
- Each value $sim_{a,b}$ should be between 0 and 1, where a higher value means that activity labels a and b are more similar.
- The value of $sim_{a,b}$ can only be equal to 1 if activity labels a and b are the same activity label.

Chapter 4

VBOPA: Variant Based Outcome Prediction Approach

This chapter describes the Variant Based Outcome Prediction Approach (VBOPA) which is, as the name already suggests, an approach that predicts the outcome of traces based on the different variants that traces can follow. More specifically, it uses only the trace variants of traces where a certain trace outcome occurs. A visual overview of the steps that compose the VBOPA can be found in Figure 4.1. The first step is concerned with some preparatory steps such as data preprocessing and data exploration and is described in Section 4.1. Then two independent paths have to be completed. The upper path is related to the definition and application of the trace similarity metric. As not all traces exactly follow an identified trace variant the trace similarity metric is defined to calculate how similar an arbitrary trace and a trace variant are. The trace similarity metric requires substitution costs for pairs of events. Sections 4.2 and 4.3 describe these two steps in detail. The lower path is concerned with the identification of all relevant trace variants that are observed within an event log and also identifies for each trace variant which traces follow that variant. Then a classifier model is trained for each identified trace variant. These two steps are explained in Sections 4.4 and 4.5. When both parallel paths are completed, Section 4.6 describes how the results of both paths are combined to perform the outcome predictions of traces based on the identified trace variants to conclude the chapter.

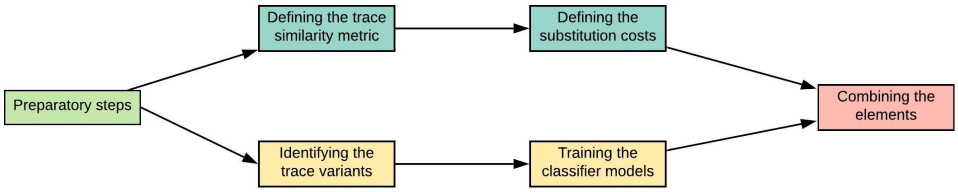


Figure 4.1: Visual representation of the Variant Based Outcome Prediction Approach (VBOPA)

4.1 Preparatory steps

A step that should always be performed when working with a data set is assessing whether data preprocessing is required. Data preprocessing is done to prevent the well-known saying *"Garbage in = garbage out* when applying data mining techniques. Data preprocessing is performed to

improve the quality of the data (thus reducing the amount of garbage), which in turn improves the quality of the obtained insights. Data preprocessing is a general term that includes several tasks, such as data integration and data cleaning. Garcia et al. [9] provide an extensive overview of the current state of data preprocessing tasks that might need to be performed on a data set. The data preprocessing tasks that are required depend on the quality of the data set at hand as each data set has its own strengths and weaknesses and therefore no specific techniques are discussed.

What is worth discussing is that Bose et al.[4] have investigated quality issues related to real-life events logs. They identified a lot of issues, related to the underlying process as well as to event data quality, and describe the impact these issues have on the application of process mining techniques. Especially event logs containing customer journey data usually face the issue that the underlying process is ill-defined (or even undefined) and are therefore very complex to analyse.

One way to gain a deeper understanding of the data set is by using generic data exploration techniques that can be applied to any data set. Behrens et al.[1] presents several techniques to investigate a data set, both via visualizations and statistical models and tests. Examples of the former are box plots and histograms, which are used to inspect the distribution of a certain feature. Examples of the latter are regression analysis and t-tests.

Another way to gain understanding of the data set is by applying process mining techniques such as process discovery. Process discovery techniques, as explained in section 2.1, can be applied to learn about the process that is executed by the traces in the data set. Leemans et al.[19] have defined an exploration cycle, which can be found in Figure 4.2. The cycle can be used to find the most suitable process model that helps to answer a specific question or hypothesis. The cycle starts with setting the scope by selecting the parameters and the perspective, after which a process model is discovered. Then this process model is evaluated and, if deemed necessary, the cycle is repeated to search for a better process model. If the most suitable process model is found, the cycle terminates and conclusions are drawn from the process model. This cycle can also be used to investigate if certain features have an impact on the process models that are discovered by adding restrictions based on feature values at the first step.

There is no single best approach for data preprocessing and data exploration and therefore all techniques mentioned in this section can be either performed, skipped or replaced by other techniques. They are simply provided to give a feeling for what this step adds to the method and provide some examples, but is in no way supposed to be complete. The way data preprocessing and data exploration is performed depends on the skills and preferences of the analyst. Also domain-specific steps, for example interviews with domain experts, that create a better understanding of a data set are performed in this step.

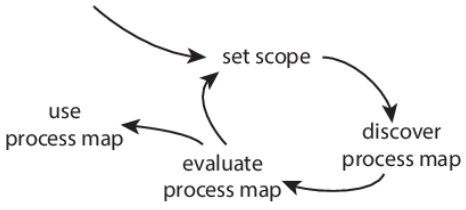


Figure 4.2: Exploration cycle as defined by Leemans et al, Figure taken from [19]

The result of this step, independent of the selected techniques, is a preprocessed data set that is understood well by the analyst. Furthermore, the data set needs to be transformed to an event log if this is not already the case.

4.2 Defining the trace similarity metric

As already mentioned in the introduction of this chapter, the trace similarity metric is used to calculate which trace variant is most similar to an arbitrary trace. The classifier model that has been trained for that trace variant can then be used to classify whether the trace outcome will occur. The technique that is selected as the basis for the trace similarity metric is the Levenshtein distance [21], which is already described in Section 2.4. The Levenshtein distance can be used to calculate the distance between two strings by calculating the minimal number of insertions, deletions and substitutions that are needed to transform one string into another string. Each operation has a default cost of 1 and the distance is then simply the sum of the costs. Even though the Levenshtein distance is used to compare two strings, it can easily be modified to make it applicable to event log data. A string is a sequence of characters, while the identified trace variants and single traces are both sequences of events. The only difference is thus that strings are composed of characters while the traces and trace variants are composed of events. The trace similarity metric (TSM) that is defined for the VBOPA is given by Formula 4.1. It is essentially the same as the measure used by Tax et al. [31], who also used the Levenshtein distance to calculate the distance between two traces. The difference is that the TSM incorporates operation costs that are based on the event log instead of the default costs of 1. The reason for introducing different costs is explained by an intuitive example after the TSM is explained in more detail.

$$TSM(t_1, t_2, SubstituteCosts) = 1 - \frac{LevenshteinDistance(t_1, t_2, SubstituteCosts)}{Max(length(t_1), length(t_2))} \tag{4.1}$$

The TSM has several properties that can be derived from Formula 4.1. First of all the outcome space is between 0 and 1 because the fraction can never be larger than 1. This is ensured by the fact that the Levenshtein distance is in the numerator while the length of the largest trace is in the denominator. Even when considering two completely dissimilar traces, which is the worst possible situation in terms of the distance calculation, the Levenshtein distance will never become greater than the length of the largest trace, provided that none of the operation costs are greater than 1. This is shown by a small example that can be found in Table 4.1, where trace 1 is transformed into trace 2 and both traces have no similar events. Four substitutions (indicated by an S) and two insertions (indicated by an I) are made to complete the transformation. Note that if trace 2 would be transformed to trace 1 the same number of operations would be required, but the two insertions are replaced by 2 deletions. Also the order of the substitutions and insertions can be changed. There is however no combination of insertions, deletions and substitutions that can transform trace 1 into trace 2 (or vice versa) with less than 6 operations and thus the Levenshtein distance between trace 1 and trace 2, assuming that all costs are equal to 1, is equal to 6. The longest trace is trace 2, whose length is equal to 6. Both the numerator and denominator of Formula 4.1 will thus be equal to 6. Consequently, the value of the TSM is equal to 0, which is ideal as the two traces are completely dissimilar. As this is the worst possible situation there is no situation possible the fraction of Formula 4.1 becomes greater than 1.

Trace 1	A	B	C	D		
Trace 2	E	F	G	H	I	J
Operation	S	S	S	S	I	I

Table 4.1: Example calculation of the Levenshtein distance between two completely dissimilar traces

Furthermore, the value of the TSM is by definition inversely related to the value of the Levenshtein distance. The higher the value of the Levenshtein distance, the lower the value of the TSM and vice versa. The reason for this is that the Levenshtein distance linearly increases when the

dissimilarity between two traces increases. Even though this property is quite straightforward, it is an important property as this property is used when the TSM is applied in the next steps. Two dissimilar traces are assumed to have a low TSM value and two similar traces are assumed to have a high TSM value. If one would like to modify the TSM by replacing the Levenshtein distance by another distance measure that breaks this linear property, it might be the case that the TSM values can not be interpreted in the same way. Replacing the Levenshtein distance with another distance measure might also break the boundary property that the TSM values will always be between 0 and 1, if the new distance measure returns values that are larger than the length of the longest trace. If that would be the case, the fraction can become greater than 1 and hence the TSM value will become negative. Such a distance measure can be selected, but then either the distance measure values need to be scaled down or the denominator value needs to be increased in order to keep the boundary property.

The properties of the TSM that are explained so far do not provide a reason why the operation costs need to be modified instead of simply using the default value of 1. This choice is motivated by another example that is shown in Table 4.2. Three traces that each consist of three events are evaluated. If the default operation costs of 1 would be used, the distance between each pair of traces would be equal to 1, as only the last event needs to be substituted for each pair of traces. However, it is not necessarily the case that events *C, D* and *E* are equally dissimilar from a contextual point of view. If the traces would originate from a data set of a health insurer, event *C* could be related to premium payments, while events *D* and *E* could both be related to insurance packages, such as informing about insurance packages and changing the insurance package. In that case it would make sense that traces 2 and 3 are more similar to each other than traces 1 and 3. This can only be achieved if the substitution costs are depending on the events that are substituted. If the cost of substituting event *E* by event *C* would be equal to 0.90 and the cost of substituting event *E* by event *D* would be equal to 0.2, the TSM values of the column on the right would be obtained. These values make sense as the substituted events are more related and thus the traces are assumed to be more similar. It is important to emphasize that this is indeed an assumption that is only met if the substitution costs of events properly represent the reality. The approaches that can be taken to define the substitution costs is discussed in Section 4.3.

Trace number	**Trace events**			**TSM(t_i, t_3)**
Trace 1	A	B	C	0.70
Trace 2	A	B	D	0.93
Trace 3	A	B	E	1.0

Table 4.2: TSM calculations when events have a different degree of similarity

4.3 Defining the substitution costs

In order to define the costs of substitution for each pair of events it needs to be known how similar each pair of events is. More specifically, for each pair of events e_1 and e_2 it should be known how similar e_1 is to e_2 and vice versa. Hence an event similarity matrix (ESM) is required: a square matrix containing values that indicate the degree of similarity between each pair of events. The structure of an ESM is already defined in Section 3.2. In the ideal scenario domain experts would define the ESM as they likely have the most knowledge about the (dis)similarity between pairs of events. However, this task can easily become impractical for a domain expert as event logs can contain a lot of different events. The amount of pairs of events for which a similarity value needs to be provided increases exponentially when the number of events increases, which is explained by Formula 4.2. Each cell in the matrix needs to be filled except the cells on the diagonal as those will always be equal to 1. This means that for an event log containing a lot of different events the task is too time-consuming to be completed by domain experts. For such event logs a data-driven

approach needs to be defined to construct the ESM.

$$NumerOfPairs = NumberOfEvents^2 - NumberOfEvents \tag{4.2}$$

Independent of the way the ESM is obtained, it needs to be transformed to a substitution cost matrix (SCM). This matrix is inverse to the ESM as events with high similarity need to have a low cost of substitution and vice versa. This is explained by Algorithm 1, where the statement in lines 1 to 5 is concerned with the approach that is taken to obtain the ESM. It is either provided by a domain expert (line 2) or it is calculated by the data-driven approach of Algorithm 2 (line 4). Either way, it is transformed to a SCM by simply subtracting all obtained values from 1. The resulting matrix can be used directly for the calculation of the Levenshtein distance between two traces. The remainder of this section describes the data-driven approach that is defined by Algorithm 2.

Algorithm 1: Calculating the substitution costs

Input : Event log L, fraction θ, Boolean *useDomainExpert*
Output: Substitution costs matrix SCM

```
1: if if useDomainExpert then
2:     ESM = domain Matrix;
3: else
4:     ESM = calculateEventSimilarityMatrix(L, θ);
5: end
6: SCM = 1 - ESM;
7: Return SCM;
```

The data-driven approach to calculate the ESM is inspired by the work of Cordewener [7] who has defined an approach to calculate an ESM from an event log by applying a variant of the market basket analysis. The work of Cordewener is already discussed in Section 2.4. The approach is adapted slightly in order to make it more general as the approach of Cordewener was tailored towards a specific event log. The approach, defined in Algorithm 2, counts how often event e_1 is directly followed by event e_2 for all pairs of events in the event log. Then this number is divided by the number of times event e_1 is observed in the entire event log. This calculation is also shown in Formula 4.3 and lines 4 to 12 of Algorithm 2 are concerned with these calculations. Note that these calculations are not performed if events e_1 and e_2 are the same event because these values are set to 1 at a later time. Therefore they are set to 0 at lines 6 and 7 to make sure that they have no impact on the calculations of the other values. For the event log used by Cordewener [7], most values in the ESM were very low and therefore a scaling step was applied to make the ESM more applicable. It can be expected that the values of the ESM obtained by lines 4 to 12 will also be low and therefore a scaling step is included. The reason is that low event similarity values have as a consequence that, if the costs of substitution are derived from the ESM, the costs of substitution will also be very close to each other. The scaling step looks at each row in the matrix and takes the largest value that is found. Then each value of the row is divided by this value multiplied by a fraction θ. This fraction is added to prevent that the largest resulting value becomes 1 as this is not allowed (see Section 3.2). The fraction θ should thus be greater than 1, but a specific value is not given as there are no clear guidelines available. It can be chosen based on intuition or by looking at the result of choosing several different values. The scaling step is shown in lines 13 to 18. Finally, the values on the diagonal need to be set to 1, as the values on the diagonal represent the similarity of an event to itself. This is shown in lines 19 to 21 of Algorithm 2.

$$Similarity(act_A, act_B) = \frac{count(act_A \rightarrow act_B)}{count(act_A)} \tag{4.3}$$

Algorithm 2: Calculating the event similarity matrix

```
1:  Function calculateEventSimilarityMatrix(L, θ);
2:  EventLabels = array of all event labels observed in L;
3:  ESM = new matrix, with EventLabels as both column and row indices;
4:  for event e1 in EventLabels do
5:      for event e2 in EventLabels do
6:          if e1 == e2 then
7:              ESM[e1, e2] = 0; // ESM[x,y] represents a specific cell of the ESM,
                    where x represents the row and y the column index
8:          else
9:              ESM[e1, e2] = Similarity(e1, e2); // Function is defined by Formula 4.3
10:         end
11:     end
12: end
13: for row in ESM do
14:     MaxValue = Max(row);
15:     for column in ESM do
16:         ESM[row, column] = ESM[row, column] / (MaxValue * θ);
17:     end
18: end
19: for event in EventLabels do
20:     ESM[event, event] = 1;
21: end
22: return ESM;
```

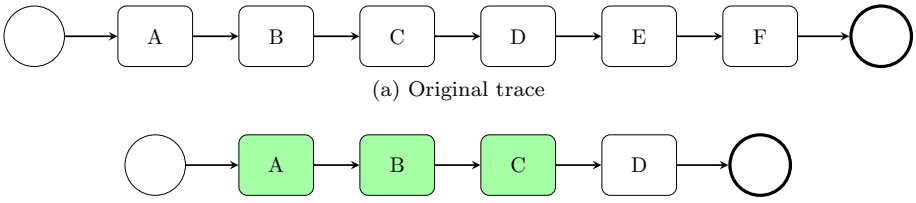
4.4 Identifying the trace variants

As already discussed in the introduction of this chapter, the VBOPA predict the trace outcome of traces based on the identified trace variants. Several trace variant definitions are found within the literature. Some definitions are based on a process model or notion of regular behavior while other definitions are based on the exact order in which events are executed. As already stated in Chapter 1, the type of traces for which the VBOPA is applicable are traces that can contain a lot of different behaviour and where it is likely that the trace outcome does not occur very often. Consequently, process models found via process discovery techniques will easily become too large and unstructured. Therefore trace variant definitions based on a process model are not deemed suitable, while a trace variant definition similar to the definition of Günther and Rozinat [12] seems more suitable. This definition does not require a process model and can thus be safely applied to an event log containing customer journey data. The definition of a trace variant that is used by the VBOPA is as follows: a process-variant is a specific sequence of events after which a certain trace outcome can be observed. This definition implies that the trace outcome is not always observed, it will become clear in the next example why this is the case. It also means that a trace variant is represented by a single sequence of events and that no events can occur after the trace outcome is observed.

An example to show how a trace variant is identified for a single trace is provided through Figure 4.3. This example also shows a specific preprocessing step that might need to be performed on the traces, which depends on the point at which the trace outcome is observed within each trace. Suppose that the trace outcome of the trace of Figure 4.3 (a) is event *D*. The trace then needs to be preprocessed into trace (b) by removing the events that occurred after event *D*. If this would not be done and trace (a) would be kept, the classifier models that are trained in a later step might lead to biased predictions. Assume that event *D* is a complaint and event *E* is an event

that only occurs after a complaint, for example a follow-up phone call. Training a classifier model on trace (a) would make the classifier fit on event E and the attributes associated to the event, for example a specific department that only performs these kind of follow-up events. This is not desirable as this means that the classifier model is trained on data that will never be observed for traces for which it needs to be predicted if a complaint occurs as then the follow-up event will definitely not have occurred yet.

After this preprocessing step the trace variant can be identified for the trace of Figure 4.3 (b), which is made clear via the colored events. The events in green are used to construct the trace variant of the trace. The last event, which is the trace outcome that needs to be predicted, is not in green and is thus not incorporated in the trace variant. The reason for this is that it is not necessarily true that this event always occurs. If event D would again be a complaint it is not hard to imagine that there will also be traces where event D does not occur. On the other hand there can definitely be traces where events A, B and C occur after each other and event D does not occur. Both traces $<A,B,C>$ and $<A,B,C,D>$ do follow the same trace variant in the context of the method. For this reason, the event that needs to be predicted is not included in the trace variant.



(a) Original trace

(b) Result when event D is the trace outcome. The events in green show the events that define the trace variant.

Figure 4.3: Log preprocessing step and trace variant identification step explained

The approach to identify the trace variants for the entire event log is explained by Algorithm 3. First a subset of the entire event log is taken by selecting all traces where the trace outcome occurs. The reason for this is that all trace variants where the trace outcome never occurs do not require a classifier model to be trained and therefore do not need to be included in the set of trace variants. Then an empty set of trace variants is initialized. Then the algorithm iterates over all traces in the subset of the event log. Each trace is preprocessed similarly to the example trace of Figure 4.3 (b) at line 4 by calling the function of Algorithm 4. Algorithm 4 simply iterates over all events of the provided trace and selects all events until the trace outcome is observed. This selection is then returned. Then the trace variant of the current trace is created at line 5 by taking the sequence of all event labels of the preprocessed trace. Then lines 6 to 8 check if the trace variant is already in the set of trace variants that are already identified. If this is not the case the trace variant is added to the set, otherwise nothing is done as this would result in duplicate values in the set of trace variants. The algorithm ends when all traces of the subset of the event log are evaluated and returns the set of identified trace variants.

Algorithm 3: Calculating the set of trace variants

Input : Event log L, trace outcome *out*
Output: List of trace variants TVS

```
1: L_outcome = subset of traces in L where out occurs;
2: TVS = new List;
3: for trace t in L_outcome do
4: |   t_prepped = preprocessTrace(t, out); // See Algorithm 4
5: |   variant = Sequence of event labels of trace t_prepped;
6: |   if variant is not in TVS then
7: |   |   TVS.append(variant);
8: end
9: Return TVS;
```

Algorithm 4: Preprocessing a trace for the process variant identification

```
1: Function preprocessTrace(t, out));
2: t_prepped = new List;
3: for event e in t do
4: |   if e is not out then
5: |   |   t_prepped.append(e)
6: |   else
7: |   |   Terminate;
8: |   end
9: end
10: return t_prepped;
```

The next step is to identify which traces follow which trace variant and which traces do not follow any trace variant. This is necessary for the classifier model training step as then it needs to be known what data has to be used for each trace variant. This step is explained via Algorithm 5 where the entire event log and the set of identified trace variants is provided. The algorithm initializes an empty list of lists equal to the length of the set of identified trace variants at line 1 to store the trace identifiers of traces that follow a trace variant. At line 2 an empty set is initialized to store all trace identifiers of traces that do not follow any trace variant. Then the algorithm iterates over all traces in the event log. Each trace is preprocessed and the sequence is retrieved. It is then checked if the sequence is equal to one of the identified trace variants. If this is the case, the unique trace identifier of the trace is added to the list at the index that represents that trace variant in line 7. If this is not the case, the trace identifier is added to the list of trace identifiers that do not follow any of the trace variants. The algorithm terminates when the entire event log is evaluated.

Algorithm 5: Identifying the traces associated to each trace variant

Input : Event log L, list of trace variants TVS and trace outcome out
Output: List of trace identifiers per variant $IDsVariant$ and list of remaining trace identifiers $IDsRemaining$

```
IDsVariant = new List of lists of length length(TVS);
IDsRemaining = new List;
for trace t in L do
    t_prepped = preprocessTrace(t, out);
    variant = Sequence of event labels of trace t_prepped;
    if variant is in TVS then
        IDsVariant[TVS[variant.index].add(t.identifier);
    else
        IDsRemaining.append(t.identifier);
    end
end
Return IDsVariant, IDsRemaining;
```

4.5 Training the classifier models

In order to train the classifier models, the traces first need to be represented as a set of features, or alternatively a number of independent variables and one dependent variable. The dependent variable is straightforward as the VBOPA aims to predict whether a trace outcome occurs. Thus, the dependent variable is a binary variable that is equal to 1 if the trace outcome occurs and is otherwise equal to 0. The set of independent variables can become as quite large as a great number of variables might be deduced from the data set. This is the point where the exploratory data analysis that is incorporated in the first step of the method (see Section 4.1) can prove its value. The exploratory data analysis could have already provided valuable insights in the characteristics of the data set in relation to the dependent variable that form the basis for the set of independent variables. However, as already mentioned in Section 2.3, it is far from trivial do decide upon the best set of independent variables due to the trade-off between including as much variables as possible and the risk of overfitting. This problem itself is an active research problem [6] and it is not the aim of this thesis assist in finding a solution. Therefore the approach to take for this step is left to the analyst that applies the VBOPA.

Nevertheless, the resulting set is assumed to have a straightforward structure as is shown in Table 4.3. Each row is an instance that can be provided to a classifier algorithm. Each instance corresponds to a single trace in the event log that is translated to a set of features. The dependent variable is the outcome class, where a value of 1 means that the outcome has occurred.

ID	Independent 1	Independent 2	...	Independent n	Outcome class
1	Value 1	Value 2	...	Value n1	0
2	Value 3	Value 4	...	Value n2	0
3	Value 5	Value 6	...	Value n3	0
4	Value 7	Value 8	...	Value n4	1
...	...	...	...	...	...

Table 4.3: Example structure of the selected set of features

Now that the traces are represented as a set of features, the classifier models for each trace variant can be trained. Classifier models are already discussed in detail in Section 2.3, where the discussed classifier models are all related to a supervised machine learning problem. Kotsiantis et al. [17] define supervised machine learning as "the process of learning a set of rules from instances

(examples in a training set), or more generally speaking, creating a classifier that can be used to generalize from new instances". In other words, a classifier model takes a set of independent variables and learns a set of rules that lead to the value of the class variable (or the dependent variable). This classifier model can then, if its performance is deemed sufficient, be used to predict the class of an instance for which the class is unknown, provided that the values for all independent variables are known. Independent of the chosen classifier model, the same steps need to be taken to properly train and evaluate the performance of any classifier model. First a train and validation set needs to be taken from the set of trace identifiers that are obtained from Algorithm 5. The train set is used to train the classifier models for each trace variant, while the validation set is kept to evaluate how well the trained classifier models perform on unseen data. The process of dividing the data in a train and validation set and the training of the classifier models is shown in Figure 4.4. On the left is the set of trace identifiers per trace variant. Each set of trace identifiers is split in a train and validation set according to a specified fraction. Usually the majority of the data is used to train the classifier model and a smaller part is used for model validation. Often 70 or 80% of the data is used for model training and 30 or 20% of the data is used for model validation. Then each train set is used to train a classifier model and the result is a set of trained classifier models for each trace variant.

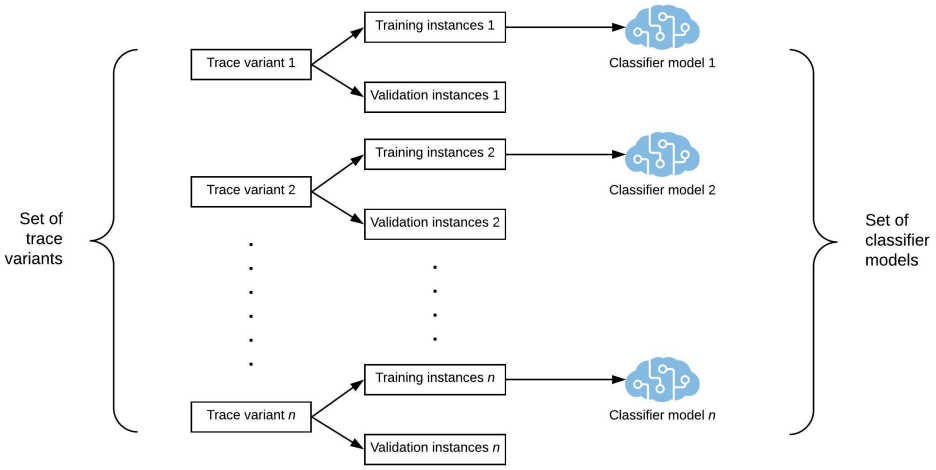


Figure 4.4: Visual overview of the training of the classifier models for each trace variant

Section 2.3 describes three types of classifier models, although obviously only one type of classifier model is required. It is also possible that one considers another type of classifier model. This step offers the possibility to evaluate multiple types of classifier models as at this point there is a train and validation set available for each trace variant. This offers the possibility to train and immediately evaluate different types of classifier models for all trace variants or even a subset of trace variants when computation time is an issue. This is very useful as it is nearly impossible to know beforehand what type of classifier model is most suitable for a data set. The choice for the classifier model is however crucial for the final performance and it is therefore recommended to take advantage of the flexibility of this step to support the decision decision of the classifier model by the data set.

4.6 Combining the elements

The final step is to combine the results of all previous steps to make the VBOPA fully applicable for the task it has been designed for: predicting the outcome of traces. This is described by

Algorithm 6, which requires the trace for which the outcome needs to be predicted, the set of trace variants (obtained in Section 4.4, the set of trained classifier models (obtained in Section 4.5 for each trace variant, the substitution cost matrix (obtained in Section 4.3 and a threshold λ.

This threshold requires some explanation as it has not been mentioned before. The threshold is used to evaluate whether the similarity between the trace and most similar trace variant is high enough to perform the prediction via the trained classifier model. If a trace is not similar enough to any of the trace variants it makes sense to conclude that it is most likely that the trace outcome will not occur and a classifier model is thus not needed for such a trace. As the trace similarity metric always returns a value will be between 0 and 1, the threshold also has to be between 0 and 1. The value λ depends heavily on the event similarity values of the ESM that is obtained from Algorithm 2 as these values impact the result of the trace similarity metric. Suppose the values of the ESM are distributed as is shown in Figure 4.5. Similarity values higher than 0.8 are not observed (when ignoring the similarity values of 1 which are on the diagonal of the ESM). Choosing a high value for λ, for example 0.8, would thus mean that it will become less likely that a trace variant will be similar enough to a trace that does not exactly follow a trace variant. This has as a consequence that for most of the traces that do not exactly follow a trace variant it will be predicted that the trace outcome does not occur. Choosing a low value of λ, for example 0.1, will have the opposite effect as this would mean that the outcome of traces that are very dissimilar to all trace variants will be predicted based on a classifier model. The parameter λ thus influences when the outcome of traces that do not exactly follow a trace variant are predicted by a classifier model and when those traces are deemed dissimilar enough to the trace variants so that no prediction is required. The choice therefore depends both on the event similarity values and the degree of dissimilarity between traces and trace variants that is acceptable.

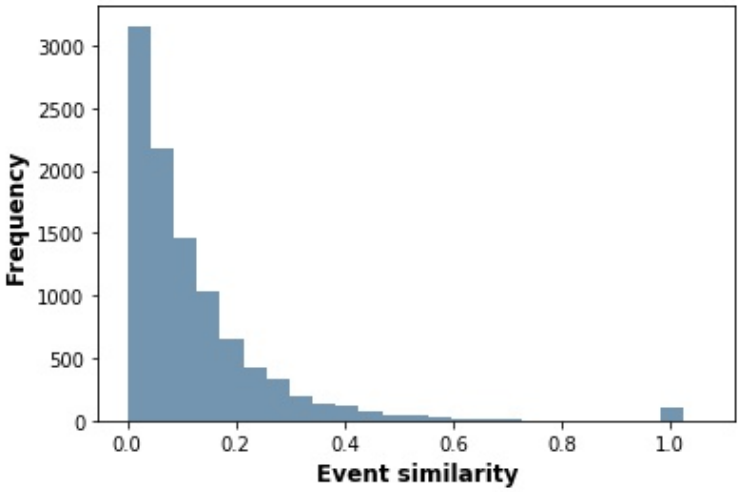


Figure 4.5: Example distribution of event similarity values

After deciding on the value of λ, Algorithm 6 can be applied to predict the outcome of an arbitrary trace. The Algorithm starts by calculating the similarity between the trace and all trace variants via the trace similarity metric, which is shown in lines 4 to 9. The trace variant that is most similar to the trace and the similarity value is then compared to the threshold that is provided in lines 10 to 13. If the similarity is above the threshold the trained classifier model of the most similar trace variant is taken from the list of trained classifier models and is used to predict whether the trace outcome will occur for the trace. If the similarity is below the threshold the prediction is set to 0. Finally, the prediction is returned.

Algorithm 6: Predicting the trace outcome of a trace by applying the VBOPA

Input : Trace tr, list of trace variants TVS, list of trained classifier models CM, substitution cost matrix SCM, threshold λ
Output: Prediction

```
1:  bestSimilarity = 0;
2:  bestVariant = -1 ;
3:  predictedValue = -1;
4:  for variant in TVS do
5:      similarity = TSM(tr, variant, SCM); // See Formula 4.1
6:      if similarity > bestSimilarity then
7:          bestSimilarity = similarity ;
8:          bestVariant = TVS[variant]; // TVS[variant] returns the index of the
                variant in TVS
9:      else
            // Do nothing
10: if bestSimilarity > λ then
11:     prediction = CM[bestVariant].predict(tr);
12: else
13:     prediction = 0;
14: Return prediction;
```

This concludes the explanation of the steps of the VBOPA and how it can be used. After completing all steps that are required to be able to perform the predictions by using Algorithm 6, the outcome of any trace can be calculated using the results. This means that a validation set can be used to evaluate the performance of the VBOPA by using Algorithm 6 to predict the outcome of each trace. The predictions can then be compared to assess how well the VBOPA is able to predict the trace outcome of traces. If it is able to this very well it could even be used in the business setting where the event data originates from to perform real time predictions.

Chapter 5

Experimental evaluation

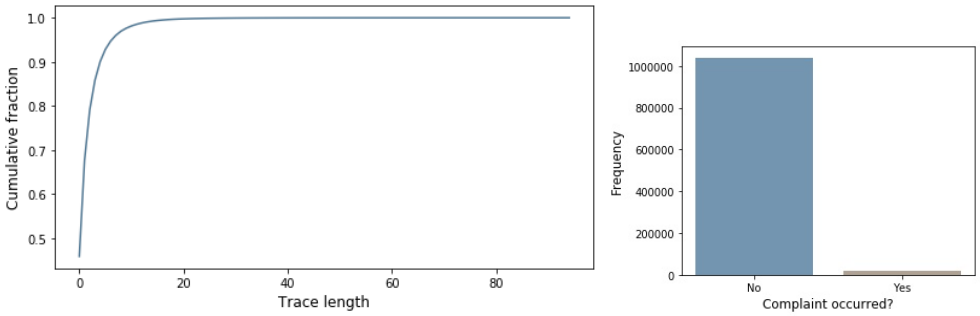
This chapter describes the application of the VBOPA on two data sets. The first data set originates from the Dutch health insurer that was already mentioned in Chapter 1. This data set has been leading in the definition of the VBOPA. The second data set is the BPI 2017 Challenge log [35] and has been selected to show that the VBOPA is not only limited to the specific domain context of the Dutch health insurer. Section 5.1 provides a description of both data sets and specifies the trace outcome that is predicted by applying the VBOPA. Section 5.2 describes how the application of the VBOPA on both data sets is evaluated by describing the performance metrics and a competitor method that the results can be compared to. Then Section 5.3 describes the actual application of the VBOPA to the Dutch health insurer data set and Section 5.4 describes the application of the VBOPA to the BPI 2017 Challenge log. Finally, Section 5.5 wraps up the experimental evaluation by drawing conclusions from the obtained results of both data sets.

5.1 Introducing the data sets

The VBOPA, which was described in Chapter 4, is applied to two data sets to evaluate how well it performs. The characteristics, challenging aspects and other relevant information of both data sets are described in this section. Section 5.1.1 describes the data set of the Dutch health insurer, Section 5.1.2 describes the BPI 2017 Challenge log. Finally, Section 5.1.3 provides a short comparison between the two data sets.

5.1.1 Dutch health insurer data set

The data set contains all contact registrations between the health insurer and its customers in 2018. Furthermore, all complaint registrations during the same time period are added, as well as some customer characteristics. The contact registrations originate from different departments and different types of communication channels are used, such as telephone, email and counter visits. The data set contains traces from just over 1 million different customers, which have had nearly 3.3 million events. Figure 5.1 (a) shows the cumulative distribution of the length of the traces in the event log. It can be observed that the majority of traces are short, as all traces that contain less than 10 events already account for over 95% of the entire data set and consequently there are little traces which contain more than 10 events. As already mentioned earlier, the goal of the health insurer is to predict whether a complaint will occur. The positive outcome is therefore that a complaint does occur. Figure 5.1 (b) shows the outcome distribution of the traces, where it can be seen that the distribution is very imbalanced: only 0.8% of the traces contain a complaint.



(a) Cumulative trace length distribution of the health insurer data set

(b) Trace outcome distribution

Figure 5.1: Characteristics of the health insurer data set visualized. Figure (a) shows a cumulative distribution of the trace length, figure (b) shows the number of traces which end with a complaint and which do not.

Several attempts have been made to mine a process model of the data set to gain insights in the process. The data is however very diverse and every attempt lead to a large and unstructured process model. The process model shown in Figure 5.2 shows the process model only for the traces that contain a complaint, which is already a very small subset of the entire data set. It is merely shown to give an intuitive feel for the complexity of the mined process model and not to inspect closely. Figure A.1 in Appendix A contains a larger version of this process model if the reader would like to inspect it in detail. This large and unstructured process model, in combination with the fact that most traces are quite short, does indicate that the process contains very diverse behaviour, which might prove to be a challenging factor.

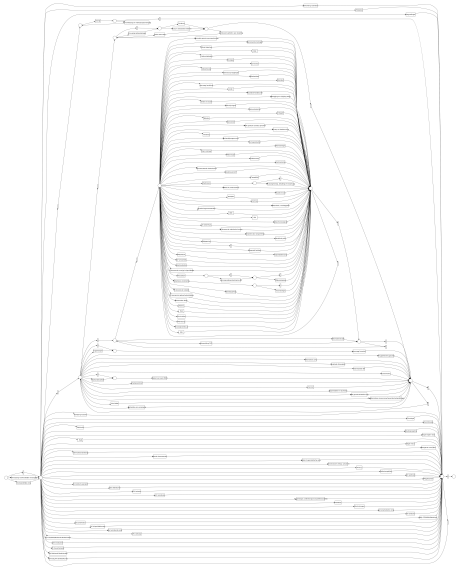


Figure 5.2: Process model of the traces that contain a complaint

5.1.2 BPI 2017 challenge log

The BPI 2017 Challenge log contains data of the loan application process at a Dutch financial institute. The log contains more than 31.000 traces that executed the loan application process,

which are composed of over 1.2 million unique events. There are three different types of events that are observed in the event log, which are the following:

- **Application:** these events are associated with the progress of the loan application.
- **Offer:** these events are associated with the offer that is created for a loan application. It is also possible that there are multiple offers present for one application.
- **Workflow:** these events are associated with the work item associated with the loan application. A work item is an object that flows through the workflow system of an organization.

A process map, created using Disco [12], can be found in Figure 5.3. Only the general flow is shown by excluding the events related to the offer and workflow. The activity and path sliders are both put at 100%, meaning the process map shows all behaviour that is observed in the event log. The process starts when a customer submits an application to the financial institute. Then, the application is checked and missing information is added, after which the financial institute provides a loan offer to the applicant. Sometimes, multiple offers are created if the first offer is not deemed satisfactory. Finally, the process ends when the application is either accepted or denied.

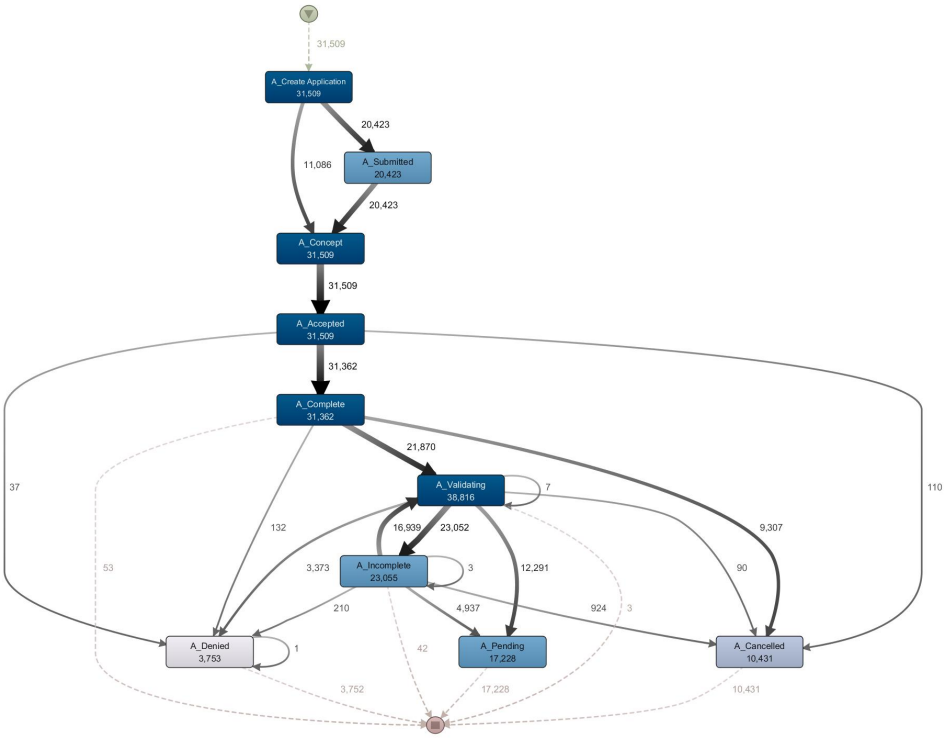
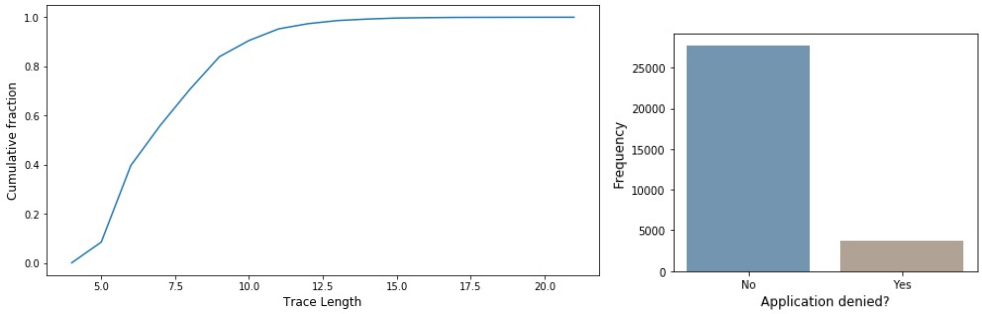


Figure 5.3: Process map of the general process flow of the BPI 2017 Challenge log

The final step of the process description provides an interesting trace outcome that can be predicted by applying the VBOPA: whether an application is denied. The reason the process is initiated is that a customer would like to receive a loan, while the product of the company is the distribution of loans. Hence both parties share the goal of successfully reaching a satisfactory offer that is acceptable for both parties. Being able to predict if an application will be denied can provide valuable insights for the business owner that can be used to improve the process in order to minimize the number of denied applications or terminate processes early to save time and

resources. The *positive* outcome in terms of the prediction task is thus a process that ends with a denied application.

The cumulative trace length distribution is shown in Figure 5.4 (a). The traces of this set are longer than the traces in the health insurer data set, although most traces are again of a similar length as around 90% of the traces contain less than 10 events. Figure 5.4 (b) shows the distribution of the trace outcome, where it can be seen that there are considerably less traces that end with a denied application than traces that do not.



(a) Cumulative trace length distribution of the BPI 2017 Challenge log

(b) Trace outcome distribution

Figure 5.4: Characteristics of the BPI 2017 Challenge log visualized. Figure (a) shows a cumulative distribution of the trace length, figure (b) shows the number of traces which end with a denied application and which do not.

5.1.3 Comparing the two data sets

There are some differences between both data sets. Generally speaking, the BPI 2017 Challenge log is less complex than the Dutch health insurer data set. This is confirmed by the fact that the process model of the BPI 2017 Challenge log is less complex while the traces contain more events. The reason for this is that the BPI 2017 Challenge log contains data from a process that is inherently structured due to a set of business rules that are in place. An example is that an application has to be created before it can be accepted or denied. This is not the case for the Dutch health insurer data set as there is no such underlying process. It is a collection of contact registrations that can be initiated both by the customers and the health insurer.

There are however also some similarities between the two data sets. Both trace outcomes occur quite infrequent, although the difference is the largest in the Dutch health insurer data set. Also, even though the trace length is longer for the traces in the BPI 2017 data set, the majority of the traces within each data set are very close to each other and a small number of traces are longer.

5.2 Evaluating the performance

This section describes how the performance of the VBOPA on both data sets will be evaluated. Firstly, the performance metrics are selected which will be used to evaluate the performance of applying the VBOPA on both data sets. The reason for selecting the performance metrics is discussed as well. Secondly, a competitor method is selected whose performance can be compared to the performance of applying the VBOPA on both data sets.

5.2.1 Selecting the performance metrics

The choice of the performance metrics that are used to evaluate the performance of applying the VBOPA is not a trivial choice. The main reason for this is that there are some characteristics of the data sets that have an influence on the choice of the performance metrics. First four performance metrics that are commonly used to evaluate the performance of classifier models are described. Then the advantages and drawbacks of the metrics are discussed in relation to the data sets.

		Predicted	
		N	P
Actual	N	TN	FP
	P	FN	TP

Table 5.1: Confusion matrix for the binary classification task

The selection of the performance metrics is partly based on the work of Gu et al. [11], who discussed the usefulness of several performance metrics when the data set is facing an unbalanced distribution of classes. The performance metrics that are selected are accuracy (Formula 5.1), F1 (Formula 5.2), precision (Formula 5.3) and recall (Formula 5.4). The four selected performance metrics are defined based on the confusion matrix, which is shown in Table 5.1. There are two possible classes: N (negative) and P (positive). In terms of the VBOPA, a positive class corresponds to an occurrence of the trace outcome and the negative class corresponds to no occurrence of the trace outcome. The confusion matrix summarizes the performance of a classifier model on a set of instances by showing how many instances of each class are correctly and incorrectly classified. The number of true positives (TP) and true negatives (TN) are concerned with the correctly classified instances, while the number of false positives (FP) and false negatives (FN) are concerned with the incorrectly classified instances.

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

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{5.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.4}$$

The reason for selecting these performance metrics is explained via an example, which is shown in Table 5.2. There are three possible confusion matrices of different classifier models given for an imaginary data set consisting of 1000 instances, of which only 100 have the positive outcome. The four performance metrics for each model are shown in Table 5.3. Model A is underfitting, which means that the model is not trying hard enough to predict the positive outcome. Model B is overfitting, meaning that the model predicts the positive outcome too easily. Model C provides a balanced classification as most instances of both outcomes are correctly classified and is therefore the preferred model. The only question that remains is which performance metrics will lead to picking model C.

A: Underfitting model

		Predicted	
		N	P
Actual	N	897	3
	P	87	11

B: Overfitting model

		Predicted	
		N	P
Actual	N	89	811
	P	0	100

C: Balanced model

		Predicted	
		N	P
Actual	N	796	104
	P	7	93

Table 5.2: Example situations showing the impact of performance measures

	A: Underfitting model	B: Overfitting model	C: Proper model
Accuracy	0.903	0.189	0.889
F1	0.193	0.198	0.626
Precision	0.786	0.110	0.472
Recall	0.11	1.0	0.93

Table 5.3: Performance of example situation

The accuracy metric, which is shown by Formula 5.1, is commonly used to evaluate the performance as it provides a single number that indicates the amount of instances that are correctly classified. The metric is simple and easy to interpret and would be perfectly fine if the data is balanced in terms of the positive and negative class, but problems arise when this is not the case. The consequence of the fact that a single number is provided is that it does not show how well each class is correctly classified. Gu et al. [11] note that this metric is therefore not appropriate when dealing with highly imbalanced data. The reason for this can be observed when comparing the accuracy scores of models A and C. Even though the number of correctly classified positive instances is a lot higher for model C, this difference can not be observed when only comparing the accuracy scores as the number of correctly classified instances is slightly lower for model C. Nevertheless, model C is preferred over model A as the positive outcome is classified much better without reaching an overfitting model such as model B. For this reason it might not be a good choice to solely use the accuracy score to evaluate the prediction performance. The recall score is based only on the data that actually belongs to the positive class (i.e. the data that is classified as TP or FN). A model that is able to correctly classify all positive outcomes will reach the highest possible value, which initially might seem ideal if the positive class does not occur very often. However, when looking at the scores of model B we can easily see that this is not true. Model B is heavily overfitting the data and therefore has a recall score of 1. This has as a consequence that most of the negative classes are also classified as a positive class which is obviously not preferred, which leads to low scores of the other performance metrics. The precision score is based only on the data that is classified as the positive class (i.e. the data that is classified as TP or TN). A model that does not incorrectly classify many negative classes as positive classes will score quite high on this metric, which is the case for model A. It can however also be seen that this has as a consequence that most positive classes are classified as the negative class as well. The precision thus has the same disadvantage as the accuracy metric.

The F1, which can be found in Formula 5.2, might prove to be the performance metric that both provides a good indication of model performance and is robust to imbalance within the data. The F1 is the harmonic mean between precision and recall and is therefore also related to how well the positive class (i.e. the infrequent class) is classified and also takes the FP and FN classifications into account. This is confirmed by the example of Table 5.3 as the F1 score of Model C is a lot higher than the F1 scores of models A and B. The F1 score is thus selected as the main performance metric for the performance evaluation. Even though it seems that the F1 score can be used as a single performance metric, the comparison of the performance metrics shows that it

is not unlikely that a single performance metric leads to an imperfect choice. Therefore all metrics will be reported for the performance evaluation, with the F1 score being the central performance metric.

5.2.2 Competitor method

A competitor method also needs to be chosen to compare the results that are obtained by applying the VBOPA. The main reason is that it is otherwise much more difficult to reason about the quality of the results and the added value that the VBOPA offers. No approach or technique has been found within the literature that is both suitable to the research problem and is available to use. Therefore a random predictor is chosen as the competitor method, which can be considered as the base line in terms of the prediction performance. It should thus definitely not be the case that the performance of the VBOPA is lower than the performance of the competitor method, as that would mean that random guessing would be a better option. On the other hand this means that the amount of performance that is gained by applying the VBOPA is closely related to the amount of variance that is explained by the VBOPA.

A random predictor is very straightforward as it is based solely on randomness. The only parameter that is required to apply the random predictor is the distribution of the trace outcome within the data set. Then the performance of the random predictor can be calculated by creating an artificial data set is created that has the same trace outcome distribution. Then a random generator can be used to assign a random number between 0 and 1 to each data point in this artificial data set. This random number is used to perform the prediction by applying the simple rule that a value smaller than 0.5 is equal to the prediction of the negative class (i.e. a value of 0), otherwise the positive class is predicted (i.e. a value of 1). An example showing what this result looks like is shown in Table 5.4. Based on the actual outcomes and the predictions, the performance metrics can be calculated. This process is repeated 1.000 times, resulting in 1.000 values for each of the performance metrics, after which the average value of each of the metrics is calculated. The obtained values are normally distributed as proven by the Central Limit Theorem [15], meaning that the average value is a valid representation of the obtained results.

Outcome	**Random number**	**Prediction**
0	0.8436	1
0	0.2646	0
0	0.5874	1
...	...	...
1	0.8459	1
1	0.1842	0

Table 5.4: Structure of the artificial data set

As is already mentioned in Section 5.1, the percentage of traces with the positive outcome is 0.8% for the health insurer data set and 11.9% for the BPI 2017 Challenge log. The calculated performance metrics for the two data sets are shown in Figures 5.5 and 5.6. As can be expected since the results are based on randomness, the accuracy and recall scores for both data sets are equal to 0.5 as each data point has a 50% chance of having its class guessed correctly. Also the low scores for the F1 and precision score can be expected as the number of true positives is much lower than the number of false positives due to the imbalance of the classes.

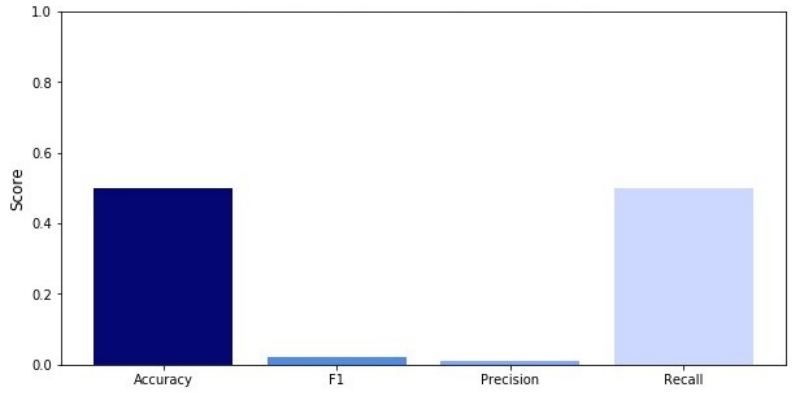


Figure 5.5: Random predictor performance for the health insurer data set

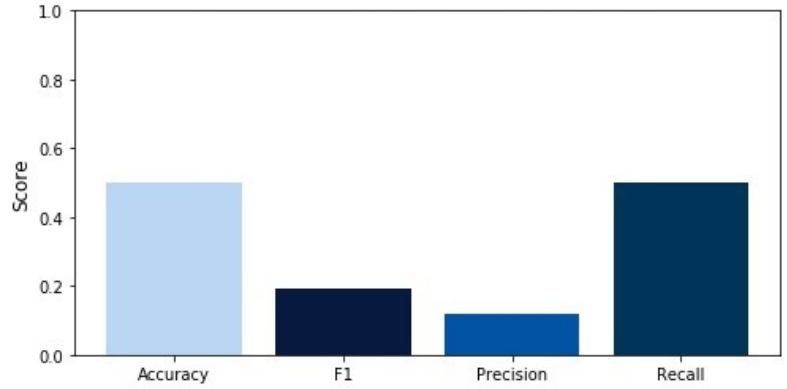


Figure 5.6: Random predictor performance for the BPI 2017 Challenge log

5.3 Applying the VBOPA to the Dutch health insurer data set

This section describes the application of the VBOPA to the data set of the Dutch health insurer which is described in Section 5.1.1. This section describes how the steps of the VBOPA are applied to the data set and shows the obtained results. Only the preparatory step is not described as this has already been done in Section 5.1.1. First Section 5.3.1 describes the calculation of the substitution costs that are required to use the trace similarity metric. Then Section 5.3.2 describes how the trace variants are identified and Section 5.3.3 describes how classifier models are trained for each trace variant. Finally the obtained results are combined and the VBOPA is applied to a validation data set to evaluate the performance in Section 5.3.4.

5.3.1 Calculating the substitution costs

The trace similarity metric (TSM) is used to calculate the similarity between traces and trace variants. Even though the TSM is identified as a step of the VBOPA, it is already defined in Section 4.2 and does not require any calculations or choices based on the data set. It does require the calculation of the substitution costs matrix SCM. The SCM depends on the event similarity matrix (ESM) which can be obtained by domain knowledge or by the data-driven approach. The event log of the health insurer contains over 90 different events and therefore the approach based

on domain knowledge is not possible. A domain expert would have to fill more than 8.000 cells according to Formula 4.2. This is obviously not feasible and therefore the data-driven approach is taken.

Algorithm 1 is used to calculate the SCM, which requires the event log, the variable indicating whether the that the data-driven method has to be applied and the fraction θ, for which the value 1.2 has been chosen. The ESM on which the SCM depends, is visualized in Figure 5.7 by showing the distribution of the event similarity values. This is done because a threshold is required in the final step which depends on this distribution. It is observed that the majority of the values are below 0.2. This large amount of pairs of events are a combination of coincidence and uncommon combinations of events. The threshold should therefore be high enough to exclude trace similarity values that include these pairs of events. The large amount of pairs of events with a low similarity value also shows that the event log contains a lot of different combinations of events and thus very diverse behaviour, which confirms the observations that were made in Section 5.1.1. Similarity values between 0.2 and 0.4 are observed fewer times and indicate the range where a big decrease of similarity values is observed. Similarity values of 0.6 or higher are rarely observed, especially when ignoring the values of 1 that represent the similarity between an event and itself. It is therefore concluded at this point that the threshold should be between 0.2 and 0.4, as this range both removes low similarity values and also does not make the threshold too strict.

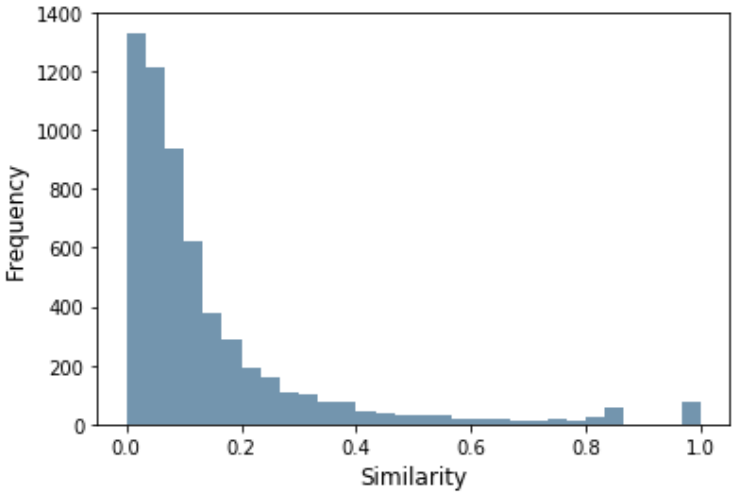


Figure 5.7: Distribution of event similarity values obtained from the health insurer event log

5.3.2 Trace variant identification

The goal of this step is to find a set of trace variants that serves as the basis for the subsequent steps. Therefore the choices made at this point are crucial for the results obtained in the end. This step results in two separate outcomes: the subset of traces of the event log that exactly follow one of the trace variants and trace representations of each of the trace variants. The first outcome is used to train the classifier models, the latter outcome is used to compare arbitrary traces to the identified trace variants.

A subset of the event log is taken to retrieve the desired set of trace variants. Only the traces for which a complaint has occurred are selected as those traces define the relevant trace variants. Algorithm 3 is then applied to this subset to identify all trace variants of this subset. The result that is obtained is thus a set of trace variants, which contains hundreds of trace variants. This

list of trace variants is then taken, as well as the entire event log, and Algorithm 5 is applied to determine for each trace in the event log if it follows any of the identified trace variants or if it does not follow any trace variant. The result is two sets containing all trace identifiers of traces that follow a trace variant and those that do not.

The result contains 220 different trace variants that are observed frequently enough in the event log. The trace variants have varying numbers of traces for which a complaint occurs: the most frequent trace variant contains over 300 traces, while some trace variants only contain 3 traces. The trace variants that are followed by less than 3 traces where the trace outcome occurs are excluded at this point. This distribution is shown in Figure 5.8, where two graphs show a histogram of the number of traces where a complaint occurs per variant. The left graph shows this histogram for all trace variants, the right graph shows this for the trace variants that contain less than 30 traces such that the large peak can be inspected in more detail. It can be seen that it is not the case that there are a few trace variants that contain a large part of the traces where a complaint occurs. In fact, only 40% of all traces where a complaint occurs are included in an identified trace variant, the remaining 60% thus follows a trace variant that is either unique or followed by only two traces. This observation implies that a very high threshold for the trace similarity values during the prediction step is not recommended as the set of trace variants does not represent enough of the trace variants of the entire event log. A high threshold would thus mean that for the majority of the remaining 60% of the traces where a complaint occurs it would be predicted that not complaint occurs.

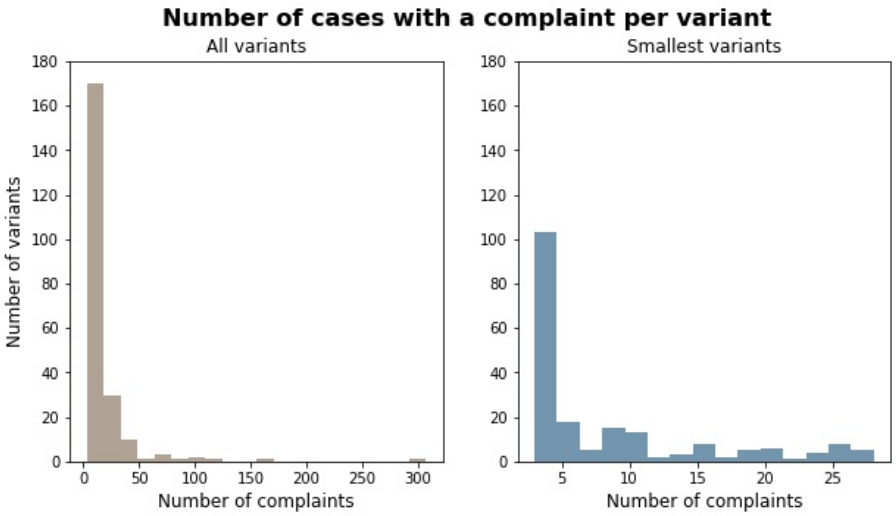


Figure 5.8: Overview of size of trace variants. The left figure shows the trace variant size distribution of all variants, the right figure shows the distribution for all trace variants consisting of less than 30 traces.

Figure 5.9 shows the trace length distribution of the identified trace variants. It can be seen that the trace variants are quite short, which could have been expected considering Figure 5.1 (a). Long traces do not occur very often within the entire event log and therefore the probability that multiple traces where a complaint occurs follow the same long trace variant is very small. Figure 5.9 shows that this indeed does not happen. This has as a consequence that long traces (i.e. traces with more than 10 events) will never be very similar to any trace variant. This is not necessarily bad as the number of traces this applies to is very small but it is still important to notice this fact.

Even though a trace variant is represented as a sequence of activity labels, it is still important to remember that a trace variant can also be represented as a process model. This is shown by

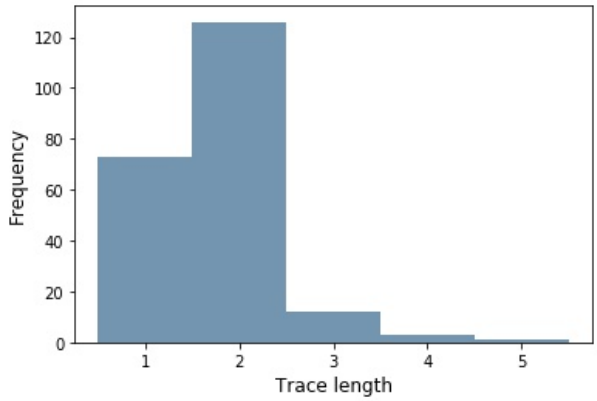


Figure 5.9: Trace length distribution of the process variants

mining the process model of the most frequent trace variant, which is shown in Figure 5.10. The process model is very simple as it consists of only two activities. It can easily be seen that there are only two possible trace execution sequences: <*Advies*> and <*Advies, Complaint*>. The place highlighted in red represents the point at which it is decided whether a complaint will occur and is thus the point for which the classifier model will be trained. This is important to realize to make sure the features are properly extracted.

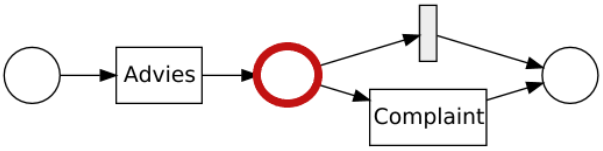


Figure 5.10: Example of a process variant. The place highlighted in red indicates where the classifier model will be trained.

5.3.3 Classifier model training

The first step to perform in order to train the classifier models for each trace variant is the selection of a set of features. It is important to remember the example process model of Figure 5.10 to make sure the extracted features do not contain data from events that occur after the decision point. The process of feature selection is, as also mentioned in Section 4.5, far from trivial and depends heavily on the skills of the analyst and the data set itself. As this process is not related to the research questions and the actual features are also not relevant for the results, this is not discussed. Then the data is split in a train and a validation set. This is done by taking the set of trace identifiers that follow the trace variants obtained earlier from Algorithm 5. The set of trace identifiers of each trace variant is split in a train and validation set. The traces of the train set are used to train the classifier models of the trace variants and the traces of the validation set are used to evaluate the final performance. To make the final performance evaluation fair, also the set of trace identifiers that do not follow a trace variant are split in a train and validation set. Only the validation set is then used for the final performance evaluation.

Even though only one type of classifier model is required to apply the VBOPA, it is investigated what type of classifier model seems most suitable. Three types of classifier models have been identified in Section 2.3: Random Forests (RF), Logistic Regression (LR) and Neural Networks (NN). These type of classifiers models have been selected because they have all proven to perform well in earlier studies and differ in terms in model interpretability. Due to calculation time issues

it is not feasible to train a classifier model for all trace variants three times. Therefore 20 trace variants have been randomly selected and for each trace variant the three different classifier models are trained. The performance of these classifiers is then evaluated, which is summarized in Figure 5.11.

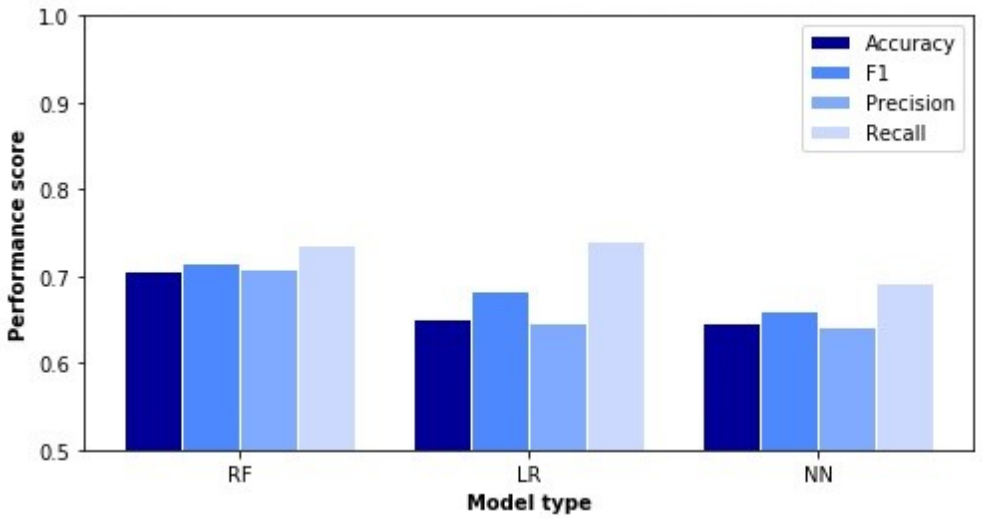


Figure 5.11: Comparing the performance of the three classifier models based on selected trace variants

It can easily be observed from Figure 5.11 that the random forest classifier is reaching the highest performance as it outperforms the other classifier models on almost all performance metrics. The most important metric, the F1 score, is clearly the highest for the random forest classifier. The difference between the three classifiers is however not very large as it is not the case that the other classifiers perform terrible in comparison to the random forest classifier. Nevertheless, the random forest classifier shows the most promising results and is thus selected to train classifier models for all trace variants.

5.3.4 Evaluating the final performance

For the final performance evaluation the results of the previous steps are combined. Before the actual performance scores can be obtained, the similarity threshold λ needs to be chosen. It was already concluded from Figure 5.7 that the most suitable threshold value will probably be between 0.2 and 0.4 due to the event similarity values of the obtained ESM. The decision also depends on the impact that the threshold has on the amount of traces that are deemed similar enough to a trace variant and those that are not. The most similar trace variant is calculated for each trace of the validation set that does not exactly follow a trace variant. The similarity values are then plotted in Figure 5.12, where it is shown how many traces would be deemed similar enough to a trace variant when increasing the threshold. If for example a threshold of 0.5 would be chosen around 11.000 traces would be deemed similar enough to a trace variant and the outcome of those traces would be predicted by a trained classifier model, while it would be predicted for the remaining traces that no complaint occurs. Based on Figures 5.7 and 5.12 it was determined that a value between 0.2 and 0.5 would be a reasonable choice. The reason for increasing the upper limit is that Figure 5.12 shows that the difference between a value of 0.4 and 0.5 is very small. The final decision is then made by selecting a number of potential threshold values. For each threshold value, a significant number of traces that are very close to that threshold value are inspected (i.e.

traces with a similarity value that is just higher or lower than the threshold value). It is then checked for those traces whether the result makes sense: are traces that are deemed similar to a trace variant indeed similar from a contextual point of view or not and vice versa. Finally, it was decided that, based on inspecting several potential threshold values, a threshold value λ of 0.4 is the best choice.

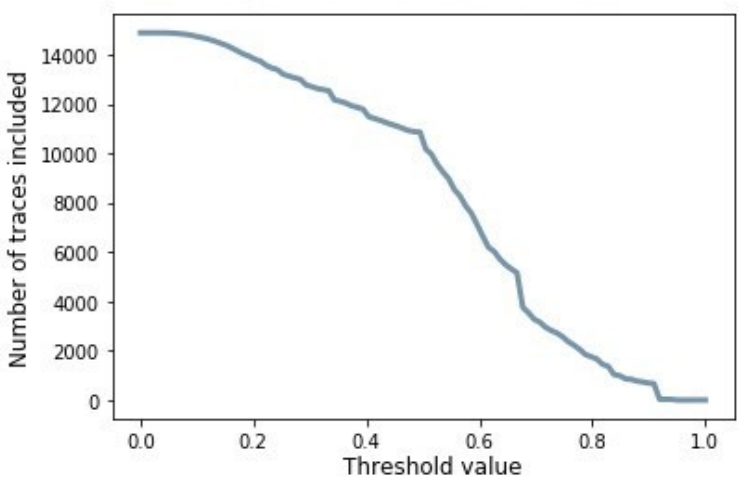


Figure 5.12: Number of traces of the validation set that are deemed similar enough to a trace variant based on the threshold value

The performance of the VBOPA is evaluated by iterating over all traces that were selected for model validation. Algorithm 6 is then applied to each trace together with the set of identified trace variants, the substitution cost matrix, the set of trained classifier models and the threshold λ of 0.4 to retrieve the predicted outcome. The result is that it is now known for each trace of the validation data what the actual trace outcome is and what trace outcome is predicted by the VBOPA. This result is used to calculate the performance metrics that were selected in Section 5.2.1 which are then compared to the performance of the random predictor of Section 5.2.2. The resulting performance values are summarized in Figure 5.13. It can easily be seen that the VBOPA reaches a higher performance than the random predictor. Especially the F1 is increased substantially. These results are discussed in more detail in Section 5.5.

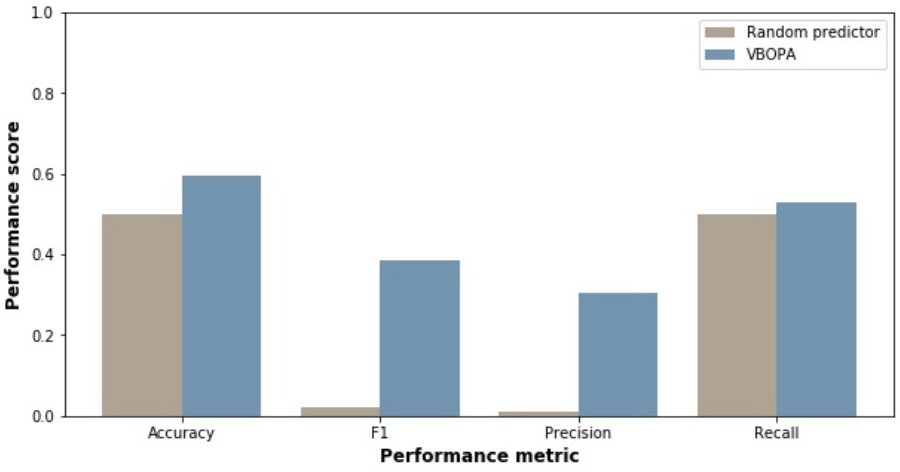


Figure 5.13: Final performance of applying the VBOPA to the health insurer data set. The performance is compared with the performance scores of the random predictor of Figure 5.5

5.4 Applying the VBOPA to the BPI 2017 Challenge log

This section describes the application of the VBOPA to the BPI 2017 Challenge log which is described in Section 5.1.2. This section describes how the steps of the VBOPA are applied to the data set and shows the obtained results. Again the preparatory step is not described as this has already been done in Section 5.1.2. First Section 5.4.1 describes the calculation of the substitution costs that are required to use the trace similarity metric. Then Section 5.4.2 describes how the trace variants are identified and Section 5.4.3 describes how the classifier models are trained for each trace variant. Finally the obtained results are combined and the VBOPA is applied to a validation data set to evaluate the performance in Section 5.4.4.

5.4.1 Calculating the substitution costs

Again, the trace similarity metric is already defined and requires no calculations or choices based on the data set. It does require a substitution cost matrix, which can be obtained by applying Algorithm 1. It needs to be decided whether the SCM is going to be defined by domain experts or by applying the data-driven approach. Since only the events of the Application type are used there are only 10 different events. This means that it would be very feasible to use domain knowledge to define the SCM as, according to Formula 4.2, this would mean that only 90 values need to be provided. This is unfortunately not possible as the data set is freely accessible from the internet and therefore no domain experts can be consulted. This is not a problem as the data-driven approach is obviously not a bad choice and it also maintains similarity between the two data sets.

Algorithm 1 is used to calculate the SCM by providing the event log L, the variable indicating that the data-driven approach needs to be used and the fraction θ. The value of 1.2 is selected, by following the same reasoning as was done for the health insurer data set. The values of the ESM are shown in Figure 5.14. It can already be seen that the distribution differs from the obtained values for the health insurer data set. One reason is that this figure is based on less data and is therefore less smooth. Furthermore a lot of similarity values are equal to 0 because there are a lot of events that never occur after each other, which can also be seen in the process map of Figure 5.3. The figure does show that the threshold that has to be chosen when performing the predictions either has to be at least equal to 0.8, as it would otherwise make very little sense.

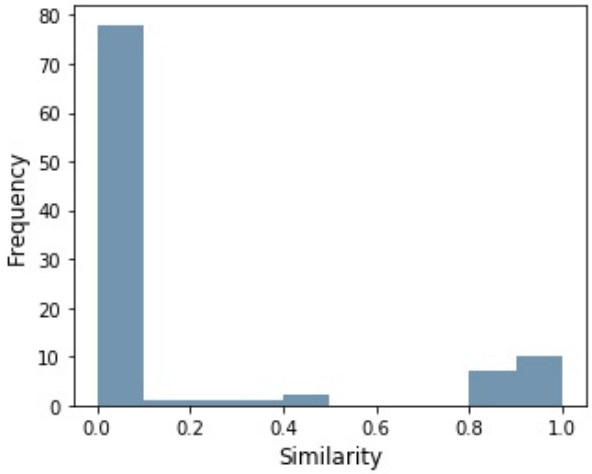


Figure 5.14: Distribution of event similarity values obtained from the BPI 2017 Challenge log

5.4.2 Trace variant identification

The goal of this step is to obtain a set of trace variants that is used to predict the outcomes of traces. Therefore only the traces of the event log L are selected that end with a denied application. Then Algorithm 3 is applied to this subset to retrieve the set of trace variants. The resulting set contains 25 different trace variants, which is considerably less than the number of trace variants that were found for the health insurer data set. Then Algorithm 5 is applied to the entire event log and the set of identified trace variants to determine which trace identifiers exactly follow an identified trace variant and which do not. Only 3 trace variants are not followed by at least 3 traces that end with a denied application and are therefore removed from the set of trace variants. The resulting 22 trace variants are visualized in Figure 5.15 where the distribution of the trace outcome is shown. It can be seen that the trace variants differ both in terms of the amount of traces that follow each trace variant and in terms of the trace outcome distribution.

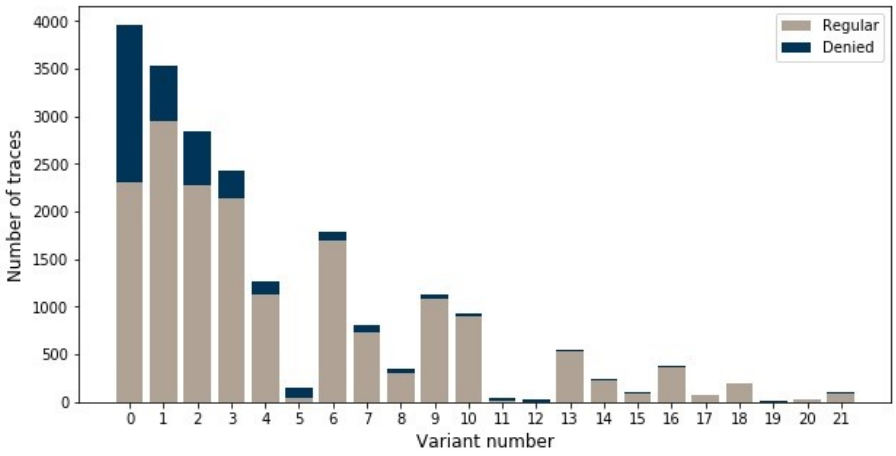


Figure 5.15: Outcome distribution per variant

5.4.3 Classifier model training

The first step is the selection of a set of features that can be used to train the classifier models. As was mentioned in Section 4.5 this is not a trivial process and depends heavily on the data set and the skills of the analyst and this process is therefore not discussed. When the features have been selected, the classifier models can be trained for each trace variant. First the set of traces of each trace variant are divided in a training set and a validation set, where the training set is used to train the classifier models and the validation set is used to evaluate the performance of the trained classifier models. Furthermore, to ensure a fair model validation process, also the set of traces that do not exactly follow a trace variant are divided similarly and only the validation data will be used for model validation.

Again several types of classifier models are compared to determine which classifier models shows the most promising results. The selected types of classifier models are again the same models as were discussed in Section 2.3: Random Forests (RF), Logistic Regression (LR) and Neural Networks (NN). These have been selected because they have shown to be suitable in previous studies and because they differ in terms of model complexity and consequently interpretability. As there are not a lot of trace variants or a large amount of data all trace variants can be used for this comparison. Figure 5.16 shows the results that are obtained by training the three types of classifier models. Based on this figure no real difference between the three types of classifier models can be observed, although the exact performance scores are indeed different from each other. Nevertheless, the difference is very small and the type of classifier model seems to be irrelevant to the prediction performance. Again, for the sake of similarity, the random forest classifier is selected.

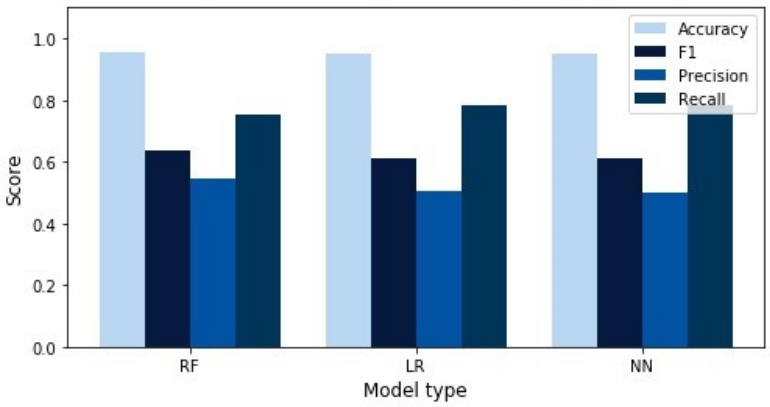


Figure 5.16: Comparing the performance of the three classifier models

5.4.4 Evaluating the final performance

The results of the previous steps are combined to enable the predictions of trace outcomes. However, the threshold λ needs to be chosen. It was already observed from Figure 5.14 that the threshold should be at least greater than or equal to 0.8. It is actually decided to set the value higher than that. The main reason is that the set of trace variants represents nearly all traces that end with a denied application. This means that all behaviour that is observed that leads to a denied application is already captured in the set of trace variants. Therefore it makes sense to be quite strict and make sure that a trace is very similar to a trace variant before even wondering whether the outcome might be a denied application. The decision is made, both due to the set of trace variants and the event similarity values, to set the parameter λ to 0.9.

The performance of applying the VBOPA is evaluated by iterating over all traces in the validation set. Algorithm 6 is then applied to each trace together with the set of identified trace variants, the substitution cost matrix, the set of trained classifier models and the threshold λ of 0.9 to retrieve the predicted outcome. The result is that it is now known for each trace in the validation set what the actual trace outcome and the predicted trace outcome by the VBOPA. This result can then be used to calculate the performance metrics that are described in Section 5.2.1 and the obtained values can be compared with the results of the random predictor that was described in Section 5.2.2. The resulting performance values are summarized in Figure 5.17. It can be seen that the VBOPA reaches a higher performance than the random predictor as all performance metrics have increased substantially.

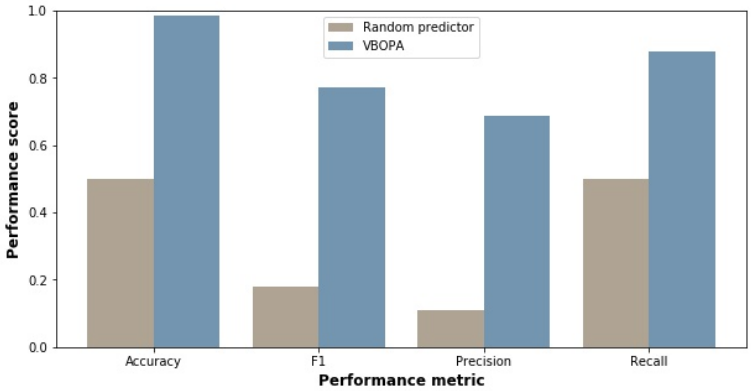


Figure 5.17: Final performance of applying the VBOPA to the BPI 2017 Challenge log. The performance is compared with the performance scores of the random predictor of Figure 5.6

5.5 Case study conclusion

The obtained results from applying the VBOPA on both data sets are now interpreted. It has been shown that the VBOPA is able to predict the trace outcomes of both data sets quite well. The results of both data sets shown a significant increase in performance compared to the competitor method. First some final remarks are made for each data set separately, then some conclusions with regards to the research questions are drawn.

The results of the health insurer data set show that the VBOPA is able to provide quite some value. It has been described that the data set is quite complex and contains a lot of varying behaviour and also contains a very large imbalance as very few traces contain a complaint. Nevertheless the VBOPA is able to reach decent performance scores. The VBOPA is able to perform significantly better than the competitor method, especially the metrics that are concerned with the prediction of the traces that do contain a complaint, the F1 and precision score, are improved the most. On a critical note, the scores are also not at the level where one would conclude that the approach is perfect. This could however not be reasonably expected, given the fact that the data set is quite complex and this is a first attempt to create a repeatable approach to perform this kind of predictions on customer journeys. It is impossible to know exactly how much performance can be actually gained, although one might get the feeling that there is some performance to be gained. In conclusion, the VBOPA provides a good approach to predict journey outcomes that are observed infrequently. It will depend on future research to determine whether the VBOPA is able to challenge new techniques.

The results of the BPI 2017 Challenge log again show the potential of the VBOPA. This data set is less challenging when compared to the health insurer data set as there is less imbalance

in terms of the trace outcome and there is less diverse behaviour. On the other hand higher performance scores are obtained for this data set. Furthermore, the results have shown that the approach is general in the sense that it can be applied to data from a different context quite easily. A limitation of this data set is that there are not a lot of different events and therefore the full potential of the trace similarity metric is not shown by this data set.

The results of both data sets show that the main research question is answered. The occurrence of complaints can be predicted based on process mining techniques by applying the VBOPA. The VBOPA makes use of trace variants and a similarity metric that is adapted from previous work in the process mining field. These elements are then combined with machine learning by including classifier models to perform predictions for each trace variant and everything is placed within the context of customer journey analysis. As already mentioned, a trace similarity metric has been defined to calculate the similarity between traces and trace variants that provides an answer to the first subquestion. The trace similarity metric depends on the Levenshtein distance and is adapted to make it more suitable to the data set by including the substitution costs based on the similarity between events. The second subquestion is answered by evaluating the performance of several three different classifier models to perform the predictions. Even though the random forest classifier was found to be the best classifier model for both data sets, it has to be noted that the performance was not an awful lot higher than the performance obtained from the other two classifier models. This was especially the case for the BPI 2017 Challenge log. It can therefore be concluded that the choice of the classifier model does indeed have some influence on the obtained results, although it can not be concluded that a certain type of classifier model drastically outperforms all other types of classifier models.

Chapter 6

Conclusions and future recommendations

This thesis proposes an approach called VBOPA that is able to predict the outcome of a data set that contains customer journey data. This approach is defined to provide a first attempt to create a repeatable approach to perform that kind of predictions on customer journey data. The approach contains three main components: a set of identified trace variants, a set of trained classifier models and a trace similarity metric. The set of trace variants is identified based on the traces that contain the outcome that is predicted by the approach. This set of trace variants can be considered as an alternative way to represent the observed behaviour that leads to the trace outcome. Then for each identified trace variant a classifier model to predict whether the trace outcome occurs is trained on the data of traces that follow that trace variant. Furthermore, a trace similarity metric is defined to compare arbitrary traces to trace variants to determine to what trace variant an arbitrary trace is most similar. These elements are then combined to predict for arbitrary traces whether the trace outcome will occur for that trace. This is done by using the trace similarity metric to determine what is the most similar trace variant and how similar the two are. If they are deemed similar enough the trained classifier model of that trace variant is used to predict whether the trace outcome occurs, if not it is determined that the trace is not similar to trace variants that lead to the trace outcome and it is predicted immediately that the trace outcome does not occur for the trace.

The VBOPA has been applied to two data sets that showed that the approach is useful for companies that possess customer journey data. Especially the application to the data set of the health insurer showed the value that the VBOPA can create. It is shown that it is able to take a very complex data set of customer journey data and predict a decent amount of traces that contain a complaint. The application to the BPI 2017 Challenge log also shows that the VBOPA is generic and can also be applied to a different context.

In terms of future research, it would make sense to investigate whether the obtained results live up to its expectations. The results seem to indicate that the approach is able to predict the outcome of traces and therefore it would make sense to investigate whether the approach can be applied in a real-life setting and still reach similar performance scores. Furthermore, there are several points within the approach where a threshold value has to be chosen in order to use the approach. It might be worth investigating what the influence of selecting different threshold values is on the final performance or if the threshold values can be chosen based on a defined approach instead of best guesses. Finally, the VBOPA is now only applied in the specific environment of a health insurer. It would be very interesting to apply the VBOPA to other data sets containing customer journey data that are similar to the health insurer data set and investigate whether

similar or even better results can be obtained and consequently show that the approach is indeed generic in the sense that it can be applied in other contexts as well.

Bibliography

[1] John T Behrens, Kristen E DiCerbo, Nedim Yel, and Roy Levy. Exploratory data analysis. *Handbook of Psychology, Second Edition*, 2, 2012. 14

[2] Gaël Bernard and Periklis Andritsos. A process mining based model for customer journey mapping. In *Forum and Doctoral Consortium Papers Presented at the 29th International Conference on Advanced Information Systems Engineering (CAiSE 2017)*, volume 1848, pages 49–56. CEUR Workshop Proceedings, 2017. 2, 5, 6

[3] Ruth N Bolton and Tina M Bronkhorst. The relationship between customer complaints to the firm and subsequent exit behavior. *ACR North American Advances*, 1995. 2

[4] RP Jagadeesh Chandra Bose, Ronny S Mans, and Wil MP van der Aalst. Wanna improve process mining results? In *2013 IEEE symposium on computational intelligence and data mining (CIDM)*, pages 127–134. IEEE, 2013. 14

[5] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 7

[6] Jie Cai, Jiawei Luo, Shulin Wang, and Sheng Yang. Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79, 2018. 21

[7] MHH Cordewener. Customer journey identification through temporal patterns and markov clustering. 2016. 9, 17

[8] Asbjørn Følstad and Knut Kvale. Customer journeys: a systematic literature review. *Journal of Service Theory and Practice*, 28(2):196–227, 2018. 5

[9] Salvador García, Julián Luengo, and Francisco Herrera. *Data preprocessing in data mining*. Springer, 2015. 14

[10] Joel Goossens, Tiblets Demewez, and Marwan Hassani. Effective steering of customer journey via order-aware recommendation. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 828–837. IEEE, 2018. 10

[11] Qiong Gu, Li Zhu, and Zhihua Cai. Evaluation measures of the classification performance of imbalanced data sets. In *International symposium on intelligence computation and applications*, pages 461–471. Springer, 2009. 29, 30

[12] Christian W Günther and Anne Rozinat. Disco: Discover your processes. *BPM (Demos)*, 940:40–44, 2012. 18, 27

[13] Christian W Günther and Eric Verbeek. Xes standard definition. *Fluxicon Process Laboratories (November 2009)*, 2014. 4, 5, 6

[14] Henrique Steinherz Hippert, Carlos Eduardo Pedreira, and Reinaldo Castro Souza. Neural networks for short-term load forecasting: A review and evaluation. *IEEE Transactions on power systems*, 16(1):44–55, 2001. 8, 9

[15] Yuzo Hosoya and Masanobu Taniguchi. A central limit theorem for stationary processes and the parameter estimation of linear processes. *The Annals of Statistics*, pages 132–153, 1982. 31

[16] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999. 6

[17] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007. 6, 21

[18] Sander JJ Leemans, Dirk Fahland, and Wil MP van der Aalst. Discovering block-structured process models from event logs-a constructive approach. In *International conference on applications and theory of Petri nets and concurrency*, pages 311–329. Springer, 2013. 4

[19] Sander JJ Leemans, Dirk Fahland, and Wil MP Van Der Aalst. Process and deviation exploration with inductive visual miner. *BPM (Demos)*, 1295(46):8, 2014. 14

[20] Katherine N Lemon and Peter C Verhoef. Understanding customer experience throughout the customer journey. *Journal of marketing*, 80(6):69–96, 2016. 1

[21] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966. 9, 15

[22] Iguácel Melero, F Javier Sese, and Peter C Verhoef. Recasting the customer experience in today's omni-channel environment. *Universia Business Review*, (50):18–37, 2016. 1

[23] Anna Meroni and Daniela Sangiorgi. *Design for services*. Routledge, 2016. 5

[24] David W Norton and B Joseph Pine. Using the customer journey to road test and refine the business model. *Strategy & Leadership*, 41(2):12–17, 2013. 5

[25] Mike Page, Leyland Pitt, and Pierre Berthon. Analysing and reducing customer defections. *Long Range Planning*, 29(6):821–834, 1996. 1

[26] Lior Rokach and Oded Maimon. Top-down induction of decision trees classifiers-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4):476–487, 2005. 7, 8

[27] Mark S Rosenbaum, Mauricio Losada Otalora, and Germán Contreras Ramírez. How to create a realistic customer journey map. *Business Horizons*, 60(1):143–150, 2017. 1

[28] Andrew I Schein and Lyle H Ungar. Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265, 2007. 7

[29] Peter Sollich and Anders Krogh. Learning with ensembles: How overfitting can be useful. In *Advances in neural information processing systems*, pages 190–196, 1996. 7

[30] Marc Stickdorn, Markus Edgar Hormess, Adam Lawrence, and Jakob Schneider. *This is service design doing: Applying service design thinking in the real world.* " O'Reilly Media, Inc.", 2018. 5

[31] Niek Tax, Natalia Sidorova, Reinder Haakma, and Wil MP van der Aalst. Event abstraction for process mining using supervised learning techniques. In *Proceedings of SAI Intelligent Systems Conference*, pages 251–269. Springer, 2016. 9, 15

[32] Alessandro Terragni and Marwan Hassani. Optimizing customer journey using process mining and sequence-aware recommendation. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 57–65. ACM, 2019. 10

[33] Wil Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*, volume 2. Springer, 2011. 4, 5

[34] Wil Van Der Aalst, Arya Adriansyah, Ana Karla Alves De Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter Van Den Brand, Ronald Brandtjen, Joos Buijs, et al. Process mining manifesto. In *International Conference on Business Process Management*, pages 169–194. Springer, 2011. 4

[35] Boudewijn van Dongen. BPI Challenge 2017, Feb 2017. 25

[36] Boudewijn F Van Dongen, Ana Karla A de Medeiros, HMW Verbeek, AJMM Weijters, and Wil MP Van Der Aalst. The prom framework: A new era in process mining tool support. In *International conference on application and theory of petri nets*, pages 444–454. Springer, 2005. 5

[37] Joost Wammes, Patrick Jeurissen, Gert Westert, and Marit Tanke. The dutch health care system. *TO REDUCE*, page 21, 2018. 1

Appendix A

Figures

This Appendix contains some figures that are not well-readable in the main text due to space limitations. In this Appendix they are enlarged to make them more readable.

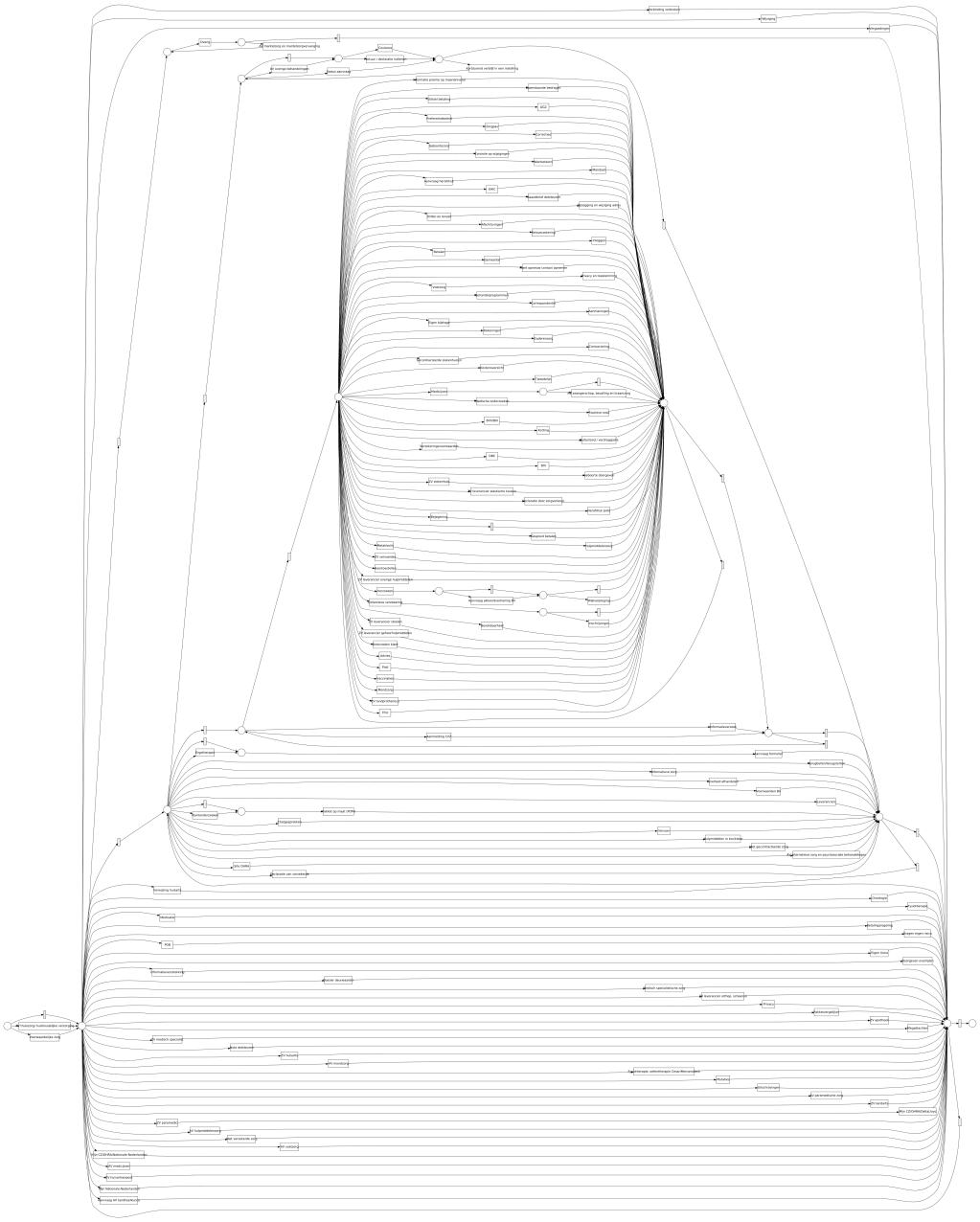


Figure A.1: Process model of the traces that contain a complaint