**Lucerne University of Applied Sciences and Arts**
**MSc Business Information Systems**

# Prediction of complaints through Machine Learning and Story Modelling

Master Thesis

Rahul Rao
rahul.rao@stud.hslu.ch

May 24, 2018

Advisors:
Prof. Dr. Alexandre de Spindler (alexandre.despindler@zhaw.ch)
Dr. Mark Cieliebak (mark.cieliebak@zhaw.ch)

# Abstract

Employees from CSS Insurance Inc. (CSS) are trained to handle daily complaints from customers with the highest quality of service. Nevertheless, the fact remains that these customers are unhappy and this reflects in the low customer satisfaction rankings CSS achieved (Comparis 2017). This research offers a possible solution to improve their ranking and to better serve their customers. The main goal was to train a model using supervised learning that is capable of predicting if a customer is likely to complain based on their previous interactions with the company. Another goal was to find out if the model would perform better if the interactions were handled in sequential order rather than a random order.

After analyzing and preprocessing the data corpus provided by CSS, the data was transformed into vectors suitable for Machine Learning (ML) with a fixed number of contact history entries per vector. These vectors were prepared using two different approaches. The first approach consisted of vectors containing three contact history entries ordered by date of contact and the second consisted of vectors containing three contact history entries randomly ordered. Multiple models were trained using different algorithmic approaches and by incrementally optimizing the data input. The suitability of these models were evaluated using precision and recall metrics. The evaluation showed that the models using naive Bayes and decision trees produced the best results. However no significant difference in precision or recall could be found between the ordered and unordered approaches, likely caused by the limited amount of contact history entries per vector.

In a final step, a complaint prediction system in the context of CSS was conceptually described. This system was designed using high level requirements specified by Fredi Bircher, a Product Owner (PO) at CSS. The goal was to design a system which could be embedded in the current landscape of CSS and which used existing processes for information flow.

This research can be considered as a first attempt at predicting potential complaints using customer interaction data. Further research and a pilot project is required before rolling out the proposed solution in a production environment.

# Acknowledgement

Machine Learning (ML) was a topic that had interested me for a while, but I just never had the opportunity or the time to get to know this topic better. Being able to learn more about ML and to gain hands-on experience, is extremely important and valuable to me. For that I would like to thank HSLU and ZHAW. Most of all, I would like to thank Alexandre de Spindler my advisor for agreeing to take on this thesis with me. His support, ideas and hints were invaluable. I would also like to thank Max Meisterhans for providing me with the necessary information on SML and for promptly answering my questions.

My special thanks go to my employer CSS for taking part in this research and for allowing me to use the data corpus. I am particularly grateful to Fredi Bircher for showing interest in this topic, for allowing me to interview him and for sharing his expertise with me.

Last but not least, I would like to express my gratitude to my parents. I can never repay them for their endless support and encouragement. I would not be where I am today without them!

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

**CCH** Customer Contact History

**CRM** Customer Relationship Management

**CSS** CSS Insurance Inc.

**DWH** Data warehouse

**ML** Machine Learning

**NLP** Natural Language Processing

**PA** Predictive Analytics

**PO** Product Owner

**SML** Story Modeller Language

**SOA** Service-Oriented architecture

**SVM** Support Vector Machine

**ZHAW** Zurich University of Applied Sciences

# 1. Introduction

This introductory chapter describes the context of this scientific research as well as its objectives and limitations.

## 1.1. Motivation

In a competitive market environment, companies seek to stand out with their products, services and innovative ideas. While increasing their sales and profits, they seek to achieve higher customer satisfaction and loyalty. In order to increase customer satisfaction, they not only need to recognise but also understand customer needs. These are big challenges for most companies. (Xu et al. 2009, p.87)

Companies may no longer just react, but must act with foresight. If not, they run the risk of being left behind by the competition. Therefore, it is important to recognize the needs and expectations of customers early or even in advance. One technical way to anticipate customer needs is Predictive Analytics (PA). PA deals with the question: "What will happen?". Based on Data Mining, ML and other methods, probabilities of future events can be provided based on historical data. (Maurer 2015) According to Gartner, PA was already on the Plateau of Productivity in 2012, moving towards the mass market (Gartner Inc. 2012). In Switzerland, PA is already being used by Swisscom in the area of the analytical CRM (Swisscom (Schweiz) AG 2014).

Companies not only need to identify the needs of their customers, but also understand them in order to be able to react on them. Natural Language Processing (NLP) offers such a possibility. NLP is a technology that aims to let people and computers interact with each other through natural language (Pustejovsky and Stubbs 2012, p.4). The combination of NLP and ML makes it possible to work through large amounts of structured and unstructured data and, at the same time,

to answer customer inquiries automatically in real time (Deloitte Consulting 2015, p.2). Forecasts from the renowned IT market research firm Gartner in 2016 show that both NLP (or Natural Language Question Answering) and ML will be ready for the mass market within two to five years (Gartner Inc. 2016). The possibilities offered by these technologies has kept the hype up for years. NLP not only offers opportunities but also brings with it its challenges. Concrete examples of these challenges are linguistic ambiguity or ontology.

At the Zurich University of Applied Sciences (ZHAW), a new machine interpretable language called Story Modeller Language (SML) is being developed based on NLP. The goal of this research is to develop a language that provides solutions to the NLP challenges mentioned above. Starting with natural language, so-called stories and their time sequences are to be mapped (Brush 2017). Furthermore, it should be possible to carry out queries in a simple manner based on the stories without having any prior programming knowledge and thus enable human-machine interaction (Spindler and Meisterhans 2017, p.4).

Of the three aforementioned technologies PA, ML and NLP, only ML is mentioned in the 2017 Gartner Hype Cycle (Gartner Inc. 2017). Nonetheless, these topics and technologies are current. For researchers and for companies, the unknown and unvetted combination of SML and ML should be interesting.

## 1.2. Problem

CSS was founded in 1899 and, with around 1.66 million policyholders, is one of the leading Swiss health, accident and property insurers (CSS Krankenversicherung AG 2017). Using their current market position they want to increase customer satisfaction significantly. According to the corporate strategy, CSS would like to be the best insurance in its sector by 2018 (CSS Krankenversicherung AG 2015).

CSS customers want the services they receive to be of high quality. They also want to be able to place their concerns knowing that they will be handled promptly and correctly. These factors have a big impact on customer satisfaction and are taken very seriously by CSS. That is why CSS attaches great importance to per-

sonal customer contact.

Personal customer contact usually takes place via service hotline employees or insurance advisors. The CSS service hotline receive on average close to 4600 phone calls a day. These employees are trained by CSS to provide the highest possible quality of service. Nevertheless, they face great challenges every day. On the one hand, they must personally be there for the customers, identify their needs, record their concerns electronically in the CRM system and process them correctly. On the other hand, they are under great pressure in terms of time and performance and therefore have to work as efficiently as possible.

In order to support these employees in their daily work and with the processing of customer concerns, organizational changes have been made. What was not considered in detail during this optimization was the actual customer contact. Depending on the mood or needs of the customer, service hotline employees have to deal differently with the customer. Since the employes do not have this information in advance, they cannot prepare for the conversation and are therefore always surprised by the customer's need. In order to be able to provide a good service, the employees must know before the contact whether the customers are satisfied or whether they will soon complain to CSS. A solution to this problem does not yet exist at CSS and should be researched as part of this master thesis.

## 1.3. Research question and objectives

ML can serve as a solution for the problem areas identified in section 1.2. Based on the concept of using sequential data to model stories in SML, the author proposes to use sequential data in ML algorithms to improve on the complaint prediction quality. Currently there are no real life systems which can transform large quantities of productive data automatically into SML, enabling users to easily make temporal queries. From these research gaps the following research question was derived, which should be answered in this thesis:

"Can customer complaints be better identified in advance if customer interactions are explicitly described in sequences rather than a disorderly collection of events?"

In order to answer this question, the following questions are additionally investigated:

- How can sequential customer interactions be brought into a form optimized for ML?
- How can customer interactions be portrayed as stories in SML?
- How can training and test datasets be assigned to the respective customer needs (complaint or not complaint)?
- How can one achieve good precision and recall values for the given requirements?
- How does a system look like that can predict in advance potential complaints from existing customer interactions?

## 1.4. Boundaries and limitations

This chapter declares the scope and limitations of this research. The areas declared as out of scope have either been excluded for organizational reasons or they have already been extensively explored or may be considered as a research gap.

- The focus is exclusively on the detection of complaints and not on other needs.
- The data provided by CSS may not be published in this work and may only be shared between CSS and the author. Furthermore it should be anonymized for data protection reasons.

- In order to keep all the data provided by CSS local, all the experiments will be conducted on the private computer of the author.

- As SML currently only supports English, the data provided by CSS may be translated into English if necessary.

- The customer interactions that are transformed into SML cannot be validated, as the test system provided by ZHAW is not yet fully functional.

- ML is implemented using different algorithmic approaches. For this research, a well-founded selection of algorithms is made. Only these will be considered and applied in the context of this research.

- The new complaint prediction system described in this research will build upon the current IT architecture of CSS and may add to the current basis.

## 1.5. Research design

This chapter presents the structure and design of this research. First, it explains on which paradigm the present work is based. Subsequently, the research process will be described. This gives an overview of the procedure and the development of the research work. Furthermore, the research methods that have been selected will be presented.

### 1.5.1. Research paradigm

Research in the field of business information systems is characterized mainly by two paradigms: behavioral science and design science (Hevner et al. 2004, p.75). This research is based on the design science approach. This approach is designed to create and evaluate IT artifacts that are intended to solve identified problems (Hevner et al. 2004, p.76). It complies with the seven guidelines of Henver (Hevner et al. 2004, p.83).

### 1.5.2. Process

The process represented in figure 1.1 depicts the defined research process. The basis on which the main part of the work, namely the experiment and the complaint prediction system design, is built, are the theoretical foundations, the state-of-the-art and the requirement. Following that, the experiment will be conducted by training a model, testing and finally evaluating it. This process is iterative and is repeated several times. The experiment should not only serve to verify the suitability of ML and sequential data for the prediction of complaints, but also provide valuable inputs for the system design. The system design should depict a possible system, embedded in the CSS context, that is able to predict customer complaints. Finally, the research will be reflected and an outlook on further possible research topics and applications will be discussed.

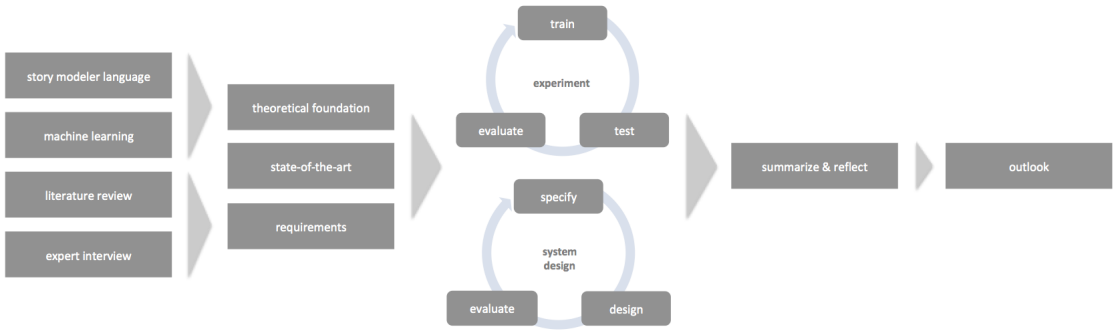


Figure 1.1.: Research process (own representation)

### 1.5.3. Method

This research is built upon four methods. In this chapter, each method and its usefulness is briefly described.

#### Literature review

In the present work a literature review according to Fettke (2006) is applied. During a literature review it is not required to review all work on a topic, but rather

only a selected few (Fettke 2006, p.258). Within the scope of this thesis, specialist literature, articles, blog entries, videos and dissertations will be considered to be relevant, should they fulfill the following criteria:

- They refer to ML and their content has not been refuted.
- They come from experts in the respective field.
- They are cited several times or recommended in forums.
- They are provided by ZHAW in relation to SML.

The results of the review are used to find out the state-of-the-art of the two subject areas SML and ML. Furthermore they are used to acquire the theoretical basis for the experiments with ML.

## Expert interview

The expert interview serves to define the requirements for the complaint prediction system design and also to evaluate it. In the following research, the interview is also used as an instrument to develop the theoretical knowledge (Lamnek and Krell 2016, p.656). The main goal is to have both the relevance and the rigor of the sytem design be judged by the expert, as well as to obtain additional inputs for its improvement and further development.

### Choice of interview method

The interview will be held with an expert. It will be conducted with the help of an interview guide (cf. Meuser and Nagel 2002). The goal is to get the expertise of an expert as effectively and efficiently as possible. Lamnek (2016) states that expert interviews should integrate both narratives as well as targeted questions. Applied to this research work, the experiences of the interviewee with the working methods of service hotline employees are discussed (narration) and also the requirements and feedback on the system design are enquired (targeted questions).

### Interviewee

At the time of this research Fredi Bircher has been selected as interview partner. Mr. Bircher works as PO at CSS and is therefore responsible for the scrum team "customer services". As PO in this area, service hotline employees are the customers of his product. Together with his team, he always strives to optimize his customers' processes and tools.

## Experiment

Following the MATTER cycle by Pustejovsky and Stubbs (2012, p.24), the experiment will be conducted several times. The corpus provided by CSS will serve as the data corpus for this experiment. The experiments always follow the same steps. First a model is trained using ML, then tested and finally measured and evaluated. The results of the experiments are compared using predefined criteria such as precision and recall. The experiments differ not only in the ML algorithms used but also in their input. The model is trained, tested and evaluated multiple times with sequential and with non-sequential data.

The experiments will help answer the research questions defined in section 1.3 and provide important insights for the system design.

## System design

The main input for the artifact to be developed will be the literature review and expert interviews. The results of the experiments will also have an influence on the design of the system. The artifact to be generated is the design of a system in the context of the CSS IT system landscape, that can predict complaints. Following the agile software development process Scrum, the artifact will be specified, designed and evaluated iteratively incrementally (Sutherland and Schwaber 2017). This approach was chosen in order to have the greatest possible flexibility and to discover serious wrong decisions as early as possible and to test alternatives (cf. Abts and Mülder 2017, p.480).

## 1.6. Outline

The research work is divided into the following seven chapters, which roughly reflect the objectives and the procedure.

*Chapter 1:* This chapter is the introduction and contains the starting point, the problem, the research question, the objective, the boundaries and the research design. The aim of this chapter is that the scope of the research work is defined, the relevance of the problem and the procedure are known to the reader.

*Chapter 2:* This chapter documents the theoretical foundations on which the work is based. Here the most important terms are defined, the technologies and existing known researches are described. The reader is helped to familiarize himself with the topic.

*Chapter 3:* In this chapter the data corpus received from CSS will be described. It will also be explained how the data will be preprocessed, mapped into vectors for ML and transformed into stories in SML. Certain research questions will be answered in this section.

*Chapter 4:* Based on the previously acquired knowledge in Chapter 2, the exact description of the experiment with regards to setup, implementation and evaluation is described in this chapter. Some of the research questions are answered here.

*Chapter 5:* The system design is described here. From requirements to the specific design and evaluation, everything is covered in this chapter. The remaining research questions are also answered here.

*Chapter 6:* This chapter focuses on a first summary of the research work. In a second part, a personal conclusion is drawn which critically reflects the results of the work.

*Chapter 7:* The final chapter gives an outlook on the potential development of the topic in research and at CSS.

# 2. Theoretical foundation

This chapter describes and defines terms that are relevant to this research. This includes the technology ML and the story modeller language SML developed by the ZHAW. The goal of this chapter is to help the reader familiarize him- or herself with the topic.

## 2.1. Machine learning

In this chapter ML, and all the relevant terms are introduced to the reader. The most well known algorithms are also described. Furthermore, in this chapter, some related works are briefly presented and commented on.

### 2.1.1. Introduction

The Oxford English dictionary defines ML as the capacity of a computer to learn from experience, i.e. to modify its processing on the basis of newly acquired information. In more detail, ML is all about automatically extracting knowledge from large quantities of data using algorithms to train a prediction model. It is at the intersection of statistics, artificial intelligence and computer science. (Müller and Guido 2016, p.1)

Manually creating expert-designed "intelligent" rule systems may be feasible for such systems where humans have a good understanding of the rules and consequences and are therefore able to derive a model from them. (Müller and Guido 2016, p.2) For other situations where the data and the rules attached to it can be considered as a black box, ML provides a good solution.

ML has found its way into our everyday lives. For instance, we get automatic recommendations on movies to watch, food to order or products to buy. Many

modern websites and devices have machine learning algorithms at their core. Facebook, Amazon and Netflix are good examples of websites using ML. (Müller and Guido 2016, p.1)

### 2.1.2. Learning methods

Depending on the feedback, learning can be classified into three categories, supervised, unsupervised and reinforcement learning.

#### Supervised learning

ML algorithms that learn from input/output pairs are called supervised learning algorithms. For each training or test result a category label or cost will be provided to the algorithm (Duda, Hart, and Stork 2000, p.16-17). The training data is made up of tuples $(x_i; y_i)$, where $x_i$ is the input and $y_i$ the corresponding target vector. Even though creating the input and output datasets are often difficult and time-consuming, supervised learning algorithms are well understood and their performance can easily be measured. An example of a supervised machine learning task is detecting fraudulent activity in credit card transactions. (Müller and Guido 2016, p.2-3)

#### Unsupervised learning

In unsupervised learning algorithms there are no labels or results for the data samples, meaning the training data comprises examples of the input vectors without any corresponding target values. In this case the algorithms usually try to cluster similar samples or to find patterns in the data (Duda, Hart, and Stork 2000, p.17). Unsupervised learning algorithms are usually harder to understand and evaluate. An example of a unsupervised machine learning task is segmenting customers into groups with similar preferences. (Müller and Guido 2016, p.3-4)

### Reinforcement learning

The goal of reinforcement learning is to find suitable actions to take in a given situation in order to maximize a reward. In this learning method the algorithm is not given an optimal output, as in supervised learning. Instead it should learn which actions it should execute by interacting with its environment. The algorithm should learn by a process of trial and error. An example where using the reinforcement learning technique makes sense is in using a neural network to learn to play a certain game such as chess or backgammon. (Bishop 2006, p.3)

Given the data corpus provided by CSS and the problem to be solved, this research will only focus on supervised learning.

## 2.1.3. Regression and Classification

In the case of supervised machine learning algorithms we can further distinguish between classification and regression tasks.

The goal in classification tasks is to predict a class label, given a set of possible labels. Classification can be further separated into *binary classification*, where the goal is to distinguish between exactly two classes and *multiclass classification*, which is classification between more than two classes. (Müller and Guido 2016, p.27)

For regression tasks, the goal is to predict a real value, a continuous number. For example predicting a person's annual income from their education, age and where they live is an example of such a regression task.

Given the objective of this research, which is to predict if a customer is going to complain or not based on past interactions, this research problem can be classified as a binary classification problem.

## 2.1.4. Algorithms

Depending on the nature of the data and the question that needs to be answered, different ML algorithms can be used. As explained in the previous chapter, this research is dealing with a binary classification problem. Using the cheat sheet

provided by the Microsoft Azure Machine Learning Team (2017) as a basis, three questions were answered and a preselection of four algorithms was made. The three questions and their answers were:

1. Is this an anomaly detection, a regression, a clustering or a classification problem?
   - It is a classification problem.
2. Is it a two- or multi-class classification problem?
   - It is a two-class or binary classification problem.
3. Are there any other additional filter criteria for the algorithm selection?
   - As mentioned in section 1.4, all the experiments must be conducted on the private computer of the author. Therefore computational limitations should be taken into account. Powerful ML platforms such as IBM Watson, Google ML, Microsoft Azure and Amazon Machine Learning, that offer great amount of computational power, a wide range of algorithms and data preparation features, cannot be used.
   - Only those algorithms can be used, that are provided by the ML library described in section 4.3.

To discuss all the possible algorithms in their feature and mathematical details would go beyond the scope of this work, so only the four selected algorithms will briefly be described here.

### Binomial logistic regression

Logistic regression is an extension of the simple linear regression. Despite its name, logistic regression is a classification algorithm and not a regression algorithm. What distinguishes the logistic from the linear regression model is that the outcome or dependent variable is binary. The formula (see equation 2.1) for logistic regression is very similar to that for linear regression, where $w$ is known as the weight or coefficient of feature $x$ and $b$ is the intercept. The difference is that for logistic regression the function does not just return the weighted sum of

the features, but thresholds the predicted value at zero. So if the value is smaller than zero, we predict one class and if it is larger than zero, then we predict the other class. (Müller and Guido 2016, p.58-59)

$$Y = w_0x_0 + w_1x_1 + w_2x_2 + ... + w_nx_n + b \tag{2.1}$$

## Naive Bayes classifier

The naive Bayes classification algorithm is based on probabilities and the theorem of Bayes, an effective formula (see equation 2.2) that can be used to calculate the probability of a posterior event based on prior knowledge (Bari, Chaouchi, and Jung 2016, p.144). *P(c | X)* represents the posterior possibility of class *c* given predictor *X*, *P(c)* represents the prior probability of class *c*, *P(X | c)* represents the likelihood of predictor *X* occurring given class *c* is true and finally *P(X)* is the probability of the predictor itself.

$$P(c|X) = \frac{P(X|c) * P(c)}{P(x)} \tag{2.2}$$

Naive Bayes classifiers are quite similar to the linear models. They are faster in training, but pay the price in providing generalization performance that is slightly worse than that of linear classifiers (Müller and Guido 2016, p.70).

## Support Vector Machine (SVM)

SVMs are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, they construct a maximum-margin separating hyperplane in a x-dimensional feature space, as depicted in figure 2.1. Using the principle of kernel substitution they are able to create general (nonlinear) models. (Brown and Mues 2012, p.3448)

Compared to other classifiers, SVMs generate stable, accurate predictions that are hardly affected by interference and are hardly prone to overfitting. SVMs are best suited for binary classification. (Bari, Chaouchi, and Jung 2016, p.142-143)

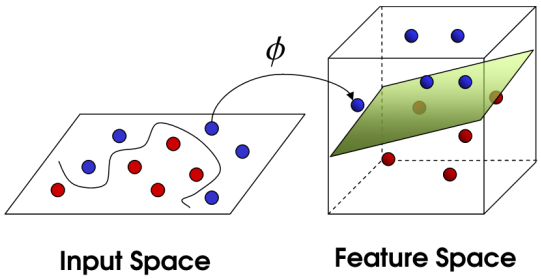


Figure 2.1.: Decision boundary found by a linear SVM (Dante 2017)

## Decision Trees

A decision tree is a predictive model, which maps observations about an item to conclusions about its target value. As modelled in figure 2.2, a decision tree consists of internal nodes that specify tests on individual input variables or attributes that split the data into smaller subsets, and a series of leaf nodes assigning a class to each of the observations in the resulting segment (Brown and Mues 2012, p.3449).

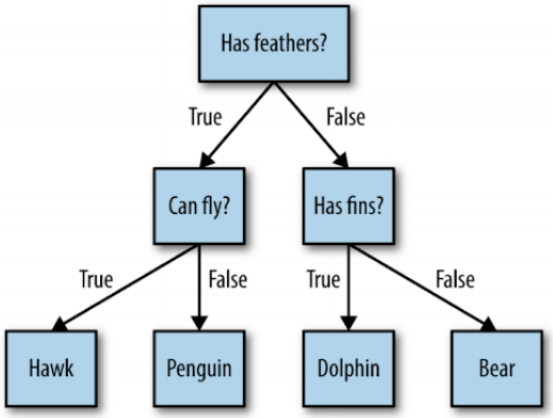


Figure 2.2.: Decision tree with nodes and leaf nodes (Müller and Guido 2016, p.73)

### 2.1.5. Related work

The binary classification problem that this thesis revolves around is not a novel problem. It has been well researched in a variety of fields.

In the field of Computer Vision, Ren and Malik (2003) published a paper where they proposed a two-class classification model for distinguishing between good and bad segmentations given an image superimposed with a human marked segmentation. To further the field of Bioinformatics, Huang and Pan (2003) proposed a two-class classification model for cancer classification. Hong, Dan and Davidson (2011) take a deep dive into social media where they investigate the problem of predicting the popularity of Twitter messages by treating their problem as a binary classification problem. The difference between these works and this research is the fact that the focus of this research is on the binary classification of *sequential* data.

In his paper *Machine Learning for Sequential Data: A Review* (2002), Thomas G. Diettrich describes six methods that have been applied to solve sequential supervised learning problems. The methods he describes use complex models like the Hidden Markov Model and Neural Networks. These models can handle sequences of different lengths. If however the sequences are of fixed length, or can be easily padded to a fixed length, they can be collapsed into a single input vector and then any of the simpler models and algorithms as described in section 2.1.4 can be used (Graves 2012).

Netzer, Lattin and Srinivasan suggest in their paper *A Hidden Markov Model of Customer Relationship Dynamics* (2007) that, using a nonhomogeneous Hidden Markov Model, one can estimate the long-term impact of customer-firm interactions on customer relationships and can dynamically segment the customer base. Similarly this research aims to develop a model that can help segment the customer base into customers that are going to complain and such that aren't. This model will also be developed using data from customer-firm interactions. However based on the limitations described in section 1.4, complex models such as the Hidden Markov Model and Neural Networks won't be used in this research.

## 2.2. Story modeler language

This chapter will briefly introduce SML and its syntax to the reader.

### 2.2.1. Introduction

SML is a new language being developed by ZHAW. It consists of a mixture of natural and formal languages. The goal of ZHAW is to create a machine interpretable language that can describe a story as a sequence of events. Furthermore SML has its own simple query language enabling people with no programming background to be able to use it. SML uses dictionaries in which words and their meanings are defined. Standard dictionaries may be extended with own specific dictionaries in order to be able to use technical or business specific words that cannot be found in the standard dictionary. Currently SML only supports the English language. (Spindler and Meisterhans 2017)

### 2.2.2. Syntax

In this chapter the syntax of SML will briefly be described. Only those feature that are relevant to this research will be presented.

#### Dictionary

A dictionary in SML is declared with the keyword DICTIONARY and its name, as represented on line 1 of listing 2.1. A dictionary can have multiple entries. Each entry is defined with the keyword ENTRY and the name of the entry, as can be seen on line 2. Each entry can have one or multiple definitions. Each definition is defined with the keyword DEFINITION, the index of the definition and its type (line 3). Following types are currently supported by SML: VERB, NOUN, ADVERB, ADJECTIVE, PREPOSITION, CONJUNCTION and NUMERAL. Each definition contains a description (line 7) and may contain multiple relationships (line 4-6). Following relationships are currently supported by SML: HYPERNYM, HYPONYM, SYNONYM and ANTONYM. (Spindler and Meisterhans 2017, p.3)

```
1 DICTIONARY "Standard English"
2     ENTRY dog
3         DEFINITION 1 NOUN
4             SYNONYM hound(1)
5             HYPONYM bloodhound(1)
6             HYPERNYM canid(1)
7             "A mammal, Canis lupus familiaris"
8         DEFINITION 2 VERB
9             "To pursue with the intent to catch"
```

Listing 2.1: Dictionary entry example

### Story

A story in SML starts with the keyword STORY and is followed by a custom name for the story as can be seen on line 1 of listing 2.2. A story must reference a dictionary that should be used or may define its own private dictionary (line 2-7). The narrative of the story is commenced with the keyword NARRATIVE (line 8). It is then followed by one to many story expressions, which can either be parallel or sequential expressions. An example for a sequential expression can be seen on line 9. A parallel expression begins with the keyword WHILE followed by a story expression (line 10-12). In a sequential expression each word is followed by a number. This number represents the definition index of the word in the dictionary. (Spindler and Meisterhans 2017, p.2)

```
1 STORY "Story of customer 0000001"
2     DICTIONARY "Private English"
3         USE "Standard English"
4         ENTRY "c0000001"
5             DEFINITION 1 NOUN
6             HYPERNYM customer(1)
7             "css customer nr. 0000001"

8     NARRATIVE
```

```
9           "c0000001"(1) drive(1) car(1)
10          WHILE {
11              "c0000001"(1) receive(1) message(2)
12          }
```

Listing 2.2: Story example

### Query

Querying stories in SML can be done in a simple way. Each query begins with the keyword QUERY. Then follows either a word with its definition index, as can be seen on line 1 of listing 2.3, or a sequential expression (line 2). A temporal meaning can be given to the query by extending it with a sequential expression that is appended to one of the following operators that are currently supported by SML: WHILE, THEN, AND and OR. (Spindler and Meisterhans 2017, p.4)

```
1 QUERY "c0000001"(1)

2 QUERY "c0000001"(1) drive(1) car(1)

3 QUERY "c0000001"(1) drive(1) car(1)
4     WHILE "c0000001"(1) receive(1) message(2)
```

Listing 2.3: Query examples

# 3. Data corpus

In this chapter the data corpus used in this research is described. The steps and the results of preprocessing the data for the experiment and its transformation into SML will be explained.

## 3.1. Customer Contact History data

The corpus provided by CSS contains the Customer Contact History (CCH) data which is generated at the point of each interaction between the customer and CSS. The corpus consists of CCH data of the last six months (between August 2017 and February 2018) from 1100 customers.

Depending on the type of interaction, for example an automatically generated policy document for the customer or a specific customer request, a CCH record is created and persisted by the responsible business domain within their own IT application. Data is shared between the IT applications of the different business domains using services deployed within the Service-Oriented architecture (SOA) landscape. Using these services, the Customer Relationship Management (CRM) system aggregates and displays the CCH data from the relevant IT applications, thereby giving service hotline employees and insurance advisors a good overview of previous interactions between the customer and CSS.

The CCH data is provided to the CRM system in the JSON data format, as can be seen in listing 3.1. This data structure was specially designed for the user interface responsible for displaying the data. Therefore duplicates and redundant information can be found in it. In the following section 3.2 this data will be preprocessed for further usage.

```
[
  {
    "leadErstelldatum": "string [format: yyyy-MM-dd]",
    "kontaktdatum": "string [format: yyyy-MM-dd]",
    "inout": {
      "value": "string",
      "code": "string",
      "message": "string",
      "name": "string",
      "text": "string"
    },
    "richtung": "enumeration",
    "kontaktkanal": "enumeration",
    "kontaktgrund": "string",
    "anliegennr": "string",
    "kontakthistoriestatus": "string",
    "verantwortlicherId": "string",
    "verantwortlicher": "string",
    "dmsDokument": {
      "dokumentNr": "string",
      "dokumentName": "string",
      "teildokumentNr": "string",
      "teildokumentTyp": "string",
      "mimeType": "string",
      "geschuetzt": false,
      "dokumentTyp": {
        "value": "string",
        "code": "string",
        "message": "string",
        "name": "string",
        "text": "string"
      },
      "dokumentSubTyp": {
        "value": "string",
        "code": "string",
        "message": "string",
        "name": "string",
        "text": "string"
      }
    },
    "vadstatus": "enumeration",
    "hasAttachment": false,
    "vectorCellId": "string",
    "postkorb": "string",
    "ersteller": "string",
    "objectId": {
      "nr": "string",
      "type": "enumeration",
      "hasLink": false
    },
    "bemerkung": "string",
    "leadname": "string",
    "leadstatus": "string",
    "leadantwort": "string",
    "beschwerde": false
  }
]
```

Listing 3.1: CSS Customer Contact History data structure

## 3.2. Preprocessing the data

As mentioned in the previous section the CCH data is aggregated from multiple sources. Because of that, the resulting JSON structure contains redundancies, entity identification issues or missing values. Furthermore the data does not only consist of continuous features, but also of categorical features such as *richtung*, *kontaktkanal*, *kontaktgrund*, *vadstatus*, *type* or *leadstatus*. As mentioned in section 1.4 there are further limitations that need to be taken into account before the data corpus can be used. In this section we will discuss the steps taken to preprocess the data corpus in order to make it useable for this research.

This process should be automated and only where required should manual steps be included. For this reason the java application *kukohi-transformator* (see Appendix C) was programmed. The steps described in the following subsections are implemented and automated within this application.

### Step 1 - Clean the data

A large amount of the data contained in the structure in listing 3.1 is not required. Some objects like *inout (line 5)*, *dokumentTyp (line 26)*, *dokumentSubTyp (line 33)* or *objectId (line 46)* contain multiple fields that were designed specifically for the UI. The properties *name* or *type* within these objects contain the relevant information, so the other properties can be eliminated.

Next, multiple properties can be merged as they are redundant. Properties like *leadErstelldatum (line 3)* and *kontaktdatum (line 4)* can be reduced to one property. *kontakthistoriestatus (line 16)* and *leadstatus (line 53)* is another example where multiple properties can be merged into one property named *status*.

Similarly *verantwortlicherId (line 17)*, *verantwortlicher (line 18)* and *ersteller (line 45)* can also be merged into one property. Their content however can vary. One property holds the name of the CSS employee, another holds the employee id and the last holds a combination of both. This is a typical example of the entity identification issue, where different systems or departments store the same data in different ways.

The whole object *dmsDokument (line 19)* is not required as its relevant informa-

tion can also be found in the property *kontaktgrund (line 14)*. It will be eliminated from the data corpus.

### Step 2 - Anonymize the data

As mentioned in section 1.4, it is required to anonymize the data for data protection reasons. Following properties from listing 3.1 need to be anonymized, using an automatically generated unique identifier:

- The employee identifier will be anonymized and stored into the new property *agent id*.
- The customer identifier will be anonymized and stored into the new property *client id*.
- The case number *(anliegennr (line 15))* will be anonymized using the prefix *cid* and stored into the new property *case id*.

### Step 3 - Filter the data

Following the two steps described above, a new JSON structure for each CCH entry is defined (see listing 3.2).

The expert interview conducted with Fredi Bircher (see Appendix A.3) also helped narrow down the list of properties that were relevant to identifying potential future complaints. Fredi hinted that the contact reason would be important. He also explained that depending on the medium the customer used to contact CSS, their expectations about response times would differ (see Appendix A.3, line 83-92). If a customer would contact CSS by letter, then he would not be expecting a response by the next day. If he however contacted CSS via the customer portal, then he would expect a response within 24 hours. So the property *medium* would be really relevant.

A further property Fredi Bircher identified as important, was the property *bemerkung* (listing 3.1, line 51). An employee could use this field to make notes about this particular case. Deriving information from this free text field could be

extremely valuable (see Appendix A.3, line 94-98). However for data protection reasons, as described in section 1.4, this data may not be used for this research.

```
[
  {
    "clientId": "string",
    "contactDate": "string [format: dd.MM.yyyy]",
    "direction": "categorical value",
    "medium": "categorical value",
    "reason": "categorical value",
    "agentId": "string",
    "caseId": "string",
    "caseType": "categorical value",
    "status": "categorical value",
    "clientAnswer": "categorical value",
    "iscomplaint": "boolean"
  }
]
```

Listing 3.2: Preprocessed Customer Contact History data structure

The JSON structure defined in the listing 3.2 above, will be used as the basis for the transformations in the following sections 3.3 and 3.4.

## 3.3. Machine learning vector transformation

Before the preprocessed data in the previous section (see listing 3.2) can be used in a supervised learning algorithm, it needs to be converted into an optimal form for ML. Firstly a vector is needed as input for the algorithm and secondly the target variable needs to be defined. These are the question which were posed in section 1.3 and which will be answered in this section. The main research question also asks to compare predictions where the customer contact history entries are treated sequentially versus predictions where the entries are in a random order. Therefore the data needs to be transformed into vectors using two different approaches.

In order to bring the customer contact history entries into a form optimized for ML, a fixed-length vector with three CCH entries per customer was defined. This decision was made considering the following:

- In order to depict a sequence, a minimum of three entries per customer is required.
- As the algorithms chosen in section 2.1.4 can only handle input of the same length and as the data corpus contains more than 350 customers with fewer than three entries, it was decided to set the length of the input vector to three entries.

To provide the target variable or label for each CCH sequence, the property *iscomplaint* of the first (newest) entry per customer will be evaluated. This entry will not be used as part of the input sequence. An overview of this transformation process can be seen in figure 3.1.

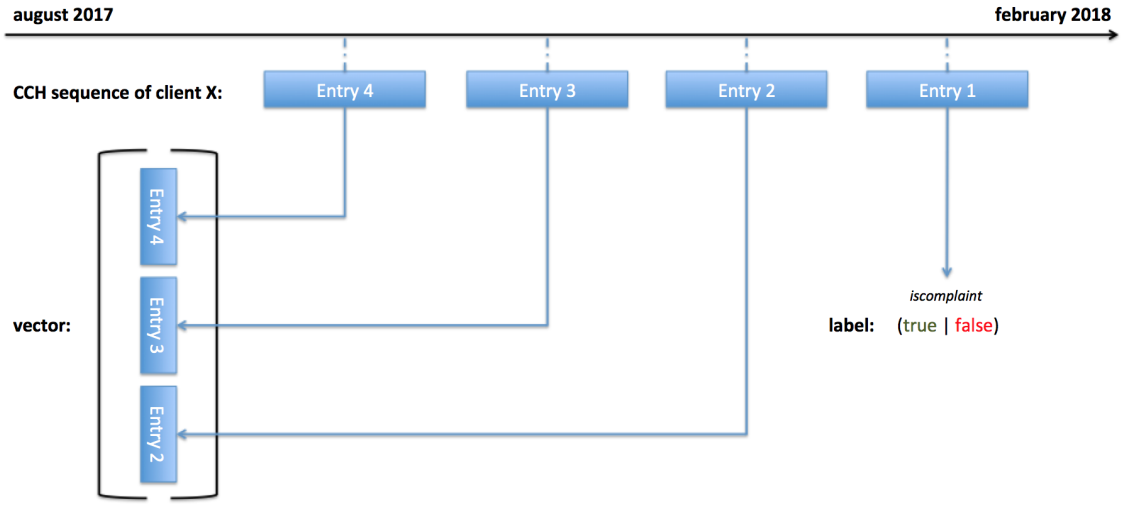


Figure 3.1.: Fixed length vector with three ordered entries per customer

The vector depicted in figure 3.1 represents the first transformation approach, where a sequence of customer contact history entries is sorted by *contactDate* in ascending order. Similarly, in the second transformation approach, a vector is created with the three entries in a random order. The vector will contain thirty

original features (ten per interaction), with each feature containing its sequence number as a suffix in its title. A list of these thirty original features can be seen in figure 3.2

```
Original features:
 ['clientId_1', 'contactDate_1', 'contactDirection_1', 'medium_1', 'reason_1',
'agentId_1', 'caseId_1', 'caseType_1', 'status_1', 'clientAnswer_1', 'clientId_2',
'contactDate_2', 'contactDirection_2', 'medium_2', 'reason_2', 'agentId_2',
'caseId_2', 'caseType_2', 'status_2', 'clientAnswer_2', 'clientId_3',
'contactDate_3', 'contactDirection_3', 'medium_3', 'reason_3', 'agentId_3',
'caseId_3', 'caseType_3', 'status_3', 'clientAnswer_3']
```

Figure 3.2.: Thirty original features provided for ML

This transformation was also automated. The output of this automation consists of two CSV files (kukohi_ml_sorted.csv, kukohi_ml_unsorted.csv) containing the data sets for the experiment. The corresponding source code can be found in the java application *kukohi-transformator* (see Appendix C).

### 3.3.1. Results of the preprocessing and transformation

Before preprocessing the data, the corpus contained CCH data for 1100 customers. Out of these 1100 data sets, 1100 vectors (one per customer) could be generated. After preprocessing and transforming the data as described in the previous sections, the number of usable vectors reduced from 1100 to 715, as can be seen in table 3.1. Out of these 715 vectors, there are 398 that lead to no complaint and 317 that lead to one. These 715 vectors will be used for the trainings and tests in chapter 4.

| Number of vectors.... | # |
|---|---|
| ... initially | 1100 |
| ... with fewer than three CCH entries | 385 |
| ... with more than three CCH entries | 715 |
| ... with at least three CCH entries before a complaint | 317 |
| ... without a complaint and at least three CCH entries | 398 |

Table 3.1.: Results of the preprocessed data corpus

## 3.4. Story modeller language transformation

In this section the question posed in section 1.3 regarding how customer interactions can be portrayed as stories in SML, will be answered. In order to transform the customer interactions into SML, the syntax described in section 2.2.2 is used. The data structure defined in listing 3.2 builds the basis for the SML transformation.

This transformation is also being automated and is currently a work in progress. The corresponding source code can be found in the java application *kukohi-transformator* (see Appendix C).

### Dictionary

Starting with the dictionary, a CSS specific one that extends the Standard English dictionary will be used as the main dictionary for each customer story. This dictionary will contain words that can be used anywhere within the context of CSS customer contact history stories, such as entries from the properties *agentId* and *reason* of listing 3.2. An example of this dictionary can be seen in the listing 3.3. The entries on line 3, 5 and 7 correspond to entries from the property *agentId* of listing 3.2. The entries on line 9 and 11 correspond to entries from the property *reason*.

```
1 DICTIONARY "CSS_Dictionary"
2     USE Standard English

3     ENTRY css
4         DEFINITION 1 NOUN "css insurance inc. the company"
5     ENTRY a0000001
6         DEFINITION 1 NOUN "css agentnr. 0000001" HYPERNYM agent
              (1)
7     ENTRY a0000002
8         DEFINITION 1 NOUN "css agentnr. 0000002" HYPERNYM agent
              (1)
9     ENTRY praemienabrechnung
```

```
10          DEFINITION 1 NOUN "bill created by css for the customer
                containing the insurance premium." HYPERNYM bill(1)
11      ENTRY "insurance cancellation"
12          DEFINITION 1 NOUN "cancellation of insurance policy with
                the company."
```

Listing 3.3: CSS dictionary example

### Story

A customer contact history story in SML also starts with the keyword STORY, followed by a custom name. In this case the custom name will be "Contact history of customer xxx", where *xxx* corresponds to the unique id of the customer. Subsequently a new dictionary is defined which extends the CSS dictionary. This dictionary will contain words and definitions specific to this story. For example the unique id of the customer or the time at which an interaction between the customer and CSS occurred, would be entries in this dictionary (see listing 3.4, lines 4-7).

```
1  STORY "Contact history of customer 0000001"
2      DICTIONARY
3          USE "CSS_Dictionary"
4          ENTRY c0000001
5              DEFINITION 1 NOUN "css customer nr. 0000001"
                   HYPERNYM customer(1)
6          ENTRY "20.10.2017 00:00"
7              DEFINITION 1 NOUN HYPERNYM datetime(1)
```

Listing 3.4: Story specific dictionary example

Following the dictionary, the narrative begins with the keyword NARRATIVE. Depending on the value of the property *direction* from listing 3.2, it can be decided who contacted whom. Using the properties *agentId* and *clientId*, the parties involved can be determined. The property *medium* contains the information about how the interaction took place. Using the information stored in the other

properties like *reason*, *status* and *iscomplaint*, the activities that take place during that customer interaction can be described, as can be seen in figure 3.3.

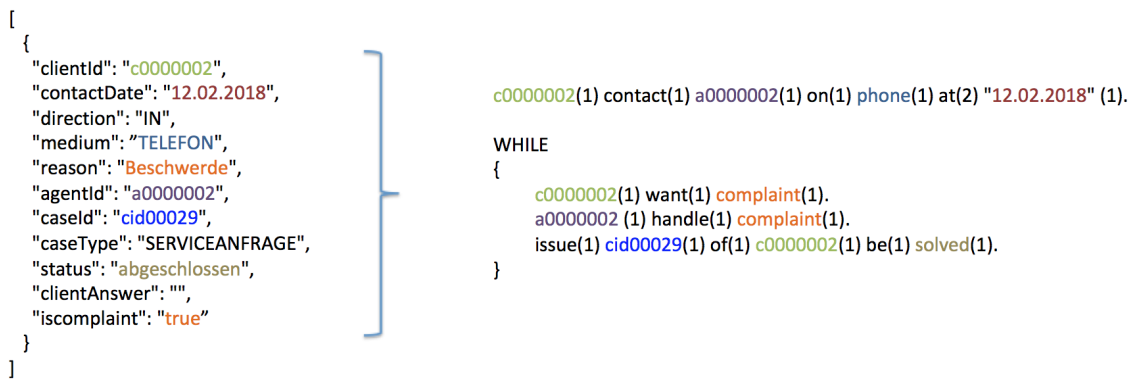


Figure 3.3.: Transformation from a customer contact history entry to SML

An example of a complete customer interaction story in SML can be found in Appendix B.1.

# 4. Experiment and results

## 4.1. Introduction

The experiment described and conducted in the following sections will help answer the main research question and further questions posed in section 1.3. Using the preprocessed data which was brought into a form suitable for ML, as described in section 3.3, multiple attempts to train models with the four chosen algorithms are made.

The goal is to achieve a good *precision* and *recall* rate for each algorithm and to compare them with each other. *Precision* measures how many of the samples predicted as positive are actually positive (see equation 4.1). *Recall* is the ratio of correctly predicted positive observations to all the observations in actual class (see equation 4.2). *Recall* can be used when it is important to avoid false negatives. As there is a trade-off between *precision* and *recall*, it is important not to only look at one of them. One way to summarize them is to also include the *f1-score* in the evaluation. The *f1-score* is the harmonic mean of *precision* and *recall* (see equation 4.3). (Müller and Guido 2016, p.284-286)

$$precision = \frac{true\ positive}{true\ positive + false\ positive} \tag{4.1}$$

$$recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{4.2}$$

$$f1\ score = 2 * \frac{precision * recall}{precision + recall} \tag{4.3}$$

The results of this experiment will not only help answer the research question, but also offer valuable input for the system design in chapter 5.

## 4.2. Process

As explained in section 1.5.2, the experiments will consist of training a model, testing it and finally evaluating its suitability for the given problem. For each of the chosen algorithms, these steps will be repeated and three individual test runs will be conducted. Before each test run the input data will be manipulated in an attempt to improve the precision and recall values. During the evaluation these precision and recall values and f1-score will be generated and discussed. In order to be able to compare results between the test runs for each algorithm, the input data for each test run for each algorithm will be the same. The preparation of this input data will be explained in section 4.4.

## 4.3. Tools

Python is the programming language of choice for this experiment, as it comes with a large array of general- and special-purpose functionality. Furthermore Python also enables creating complex graphical user interfaces and web services, allowing for integration into existing systems. In this section a short overview of all the frameworks, tools and libraries will be presented.

### Anaconda

Anaconda is a Python distribution made for large-scale data processing, predictive analytics and scientific computing. It comes with NumPy, SciPy, matplotlib, scikit-learn, pandas and Jupyter Notebook. (Müller and Guido 2016, p.6)

### Jupyter Notebook

The Jupyter Notebook is an interactive environment for running code in the browser. Data scientists widely use this as it is a great tool for data analytics. (Müller and

Guido 2016, p.7)

### NumPy

NumPy is one of the fundamental packages for scientific computing in Python. It contains functionality for multidimensional arrays and high-level mathematical functions. (Müller and Guido 2016, p.7)

### matplotlib

matplotlib is the main scientific plotting library in Python. With it one can make great visualisations such as line charts, histograms, scatter plots, and so on. (Müller and Guido 2016, p.9)

### scikit-learn

scikit-learn is a tool for data mining and data analysis. It is built on top of NumPy, SciPy, and matplotlib. It contains a number of state-of-the-art machine learning algorithms, which will be used during this research.

### pandas

pandas is a Python library for data manipulation and analysis. It offers a data structure called the DataFrame. The pandas DataFrame is a table, similar to an Excel spreadsheet. It offers a wide range of functionality to modify and operate on the table. Another functionality it offers is the ability to ingest data from a great variety of file formats and databases, like SQL, Excel and CSV files. (Müller and Guido 2016, p.10)

#### Versions

Following versions of the aforementioned libraries are used in this research:

| library | version |
|---|---|
| Python | 3.6.3 |
| pandas | 0.20.3 |
| matplotlib | 2.1.0 |
| NumPy | 1.13.3 |

Table 4.1.: Versions of libraries used

Finally, as explained in section 1.4, all the experiments will be conducted on the author's private computer, an early 2015 model MacBook Pro with a 2.7 GHz Intel Core i5 processor, 8GB RAM and an Intel Iris Graphics 6100 GPU.

## 4.4. Test setup

The initial setup for the test runs with each algorithm is the same. The pre-processed data is stored in two CSV files, as mentioned in section 3.3. One file contains the data where the vectors contain customer contact history entries in ascending order and the other file contains the data where the first 75% of the vectors contain entries in a random order and the remaining 25% are in in ascending oder. The 25% is to be used for training purposes and in order to be able to compare the results between the two models. These two files are the original input for the machine learning algorithms.

### Preparing the data

#### Data for test run nr. 1

As can be seen in listing 4.1 on *line 1*, the data is imported from the CSV file using the functionality provided by the pandas library. The imported data is stored into a DataFrame. On *line 2* and *3* the content of the original DataFrame *df* is copied

into two separate DataFrames, one containing all the rows only with the features (*df_features*) and one containing all the rows only with the labels (*df_labels*). This results in 30 features and 1 label per vector.

```
1 df = pd.DataFrame.from_csv('input-data/kukohi_ml_sorted.csv',
      index_col=None, encoding='UTF-8', sep=';')

2 df_features = df.iloc[:,0:30]
3 df_labels = df.iloc[:,-1]

4 df_onehotencoded_features = pd.get_dummies(df_features)
```

Listing 4.1: Preparing data for test run nr. 1

The input or predictant variable needs to be either binary or an ordinal variable. This means that the categorical variables stored in the DataFrame, will have to be converted into ordinal variables. This can be done using the process of One hot encoding. One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. Each categorical feature is expanded into one new feature for each possible value. (Müller and Guido 2016, p.217-218)

As can be seen in listing 4.1 on *line 4*, the function *get_dummies()* provided by the pandas library is used to automatically one hot encode the relevant features from the original DataFrame *df_features*. The number of features after this process is complete, grows significantly from 30 to 4298. This is the case, as each an every feature from the original DataFrame is interpreted as a categorical variable. So for each *clientId*, *agentId* and *caseId* an independent feature is created, causing the number of features to increase to 4298.

#### Data for test run nr. 2

Looking at the one hot encoded data set created for the first test run, it is evident that a lot of additional features are being generated because the *clientId*, *agentId* and *caseId* are being treated as categorical values. In order to change that and in order to keep the defined sequence of entries in the vector, the original DataFrame

*df* is split into three DataFrames of equal length (see listing 4.2, *lines 2, 11, 20*), each containing all the properties of one customer contact history entry.

For each of these DataFrames, the *clientId-*, *agentId-* and *caseId* strings are converted into numerical values of type int (see *lines 4, 7, 9*). Only then is the DataFrame one hot encoded (see *lines 10, 19, 28*) and finally merged into one DataFrame (see see *line 29*). In comparison to the first run, the number of features has reduced significantly from 4298 to 694.

```
1  df = pd.DataFrame.from_csv('input-data/kukohi_ml_sorted.csv', index_col=None, encoding='UTF-8', sep=';')
2  df_features_temp1 = df.iloc[:,0:10]
3  df_features_temp1['clientId_1'] = df_features_temp1['clientId_1'].str.replace('c','1')
4  df_features_temp1['clientId_1'] = pd.to_numeric(df_features_temp1.clientId_1, errors='coerce')
5  df_features_temp1['agentId_1'] = df_features_temp1['agentId_1'].str.replace('a','3')
6  df_features_temp1['agentId_1'] = df_features_temp1['agentId_1'].str.replace('css','399999')
7  df_features_temp1['agentId_1'] = pd.to_numeric(df_features_temp1.agentId_1, errors='coerce')
8  df_features_temp1['caseId_1'] = df_features_temp1['caseId_1'].str.replace('cid','5')
9  df_features_temp1['caseId_1'] = pd.to_numeric(df_features_temp1.caseId_1, errors='coerce').fillna(0).astype(np.int64)
10 df_onehotencoded_temp1 = pd.get_dummies(df_features_temp1)

11 df_features_temp2 = df.iloc[:,10:20]
12 df_features_temp2['clientId_2'] = df_features_temp2['clientId_2'].str.replace('c','1')
13 df_features_temp2['clientId_2'] = pd.to_numeric(df_features_temp2.clientId_2, errors='coerce')
14 df_features_temp2['agentId_2'] = df_features_temp2['agentId_2'].str.replace('a','3')
15 df_features_temp2['agentId_2'] = df_features_temp2['agentId_2'].str.replace('css','399999')
16 df_features_temp2['agentId_2'] = pd.to_numeric(df_features_temp2.agentId_2, errors='coerce')
17 df_features_temp2['caseId_2'] = df_features_temp2['caseId_2'].str.replace('cid','5')
18 df_features_temp2['caseId_2'] = pd.to_numeric(df_features_temp2.caseId_2, errors='coerce').fillna(0).astype(np.int64)
19 df_onehotencoded_temp2 = pd.get_dummies(df_features_temp2)

20 df_features_temp3 = df.iloc[:,20:30]
21 df_features_temp3['clientId_3'] = df_features_temp3['clientId_3'].str.replace('c','1')
22 df_features_temp3['clientId_3'] = pd.to_numeric(df_features_temp3.clientId_3, errors='coerce')
23 df_features_temp3['agentId_3'] = df_features_temp3['agentId_3'].str.replace('a','3')
24 df_features_temp3['agentId_3'] = df_features_temp3['agentId_3'].str.replace('css','399999')
25 df_features_temp3['agentId_3'] = pd.to_numeric(df_features_temp3.agentId_3, errors='coerce')
26 df_features_temp3['caseId_3'] = df_features_temp3['caseId_3'].str.replace('cid','5')
27 df_features_temp3['caseId_3'] = pd.to_numeric(df_features_temp3.caseId_3, errors='coerce').fillna(0).astype(np.int64)
28 df_onehotencoded_temp3 = pd.get_dummies(df_features_temp3)

29 df_features = pd.concat([df_onehotencoded_temp1, df_onehotencoded_temp2, df_onehotencoded_temp3], axis=1)
30 df_labels = df.iloc[:,-1]

31 logreg = LogisticRegression()
32 logreg.fit(X_train, y_train)
33 pred_logreg = logreg.predict(X_test)
34 print(classification_report(y_test, pred_logreg))
```

Listing 4.2: Preparing data for test run nr. 2

### Data for test run nr. 3

For the final test run, the features will be further optimized. In the previous run, the data set still consisted of 694 features. This is mainly caused by the *contactDate* feature, which is treated as a categorical variable. If the *contactDate* would consist

of only a few possible values, it would make sense to leave it as a categorical variable. But as any date is possible, it should be handled in a different way.

The Python Date class offers the function *toordinal()* which returns a proleptic Gregorian ordinal of the date. This data manipulation will be done in addition to the manipulation done for the second run, as can be seen in listing 4.3 on *lines 11, 22, 33*. After transforming the *contactDate* and one hot encoding and merging the data sets, the number of features further reduce from 694 to 391. Now there are no features unnecessarily being treated as categorical.

```
1  df = pd.DataFrame.from_csv('input-data/kukohi_ml_sorted.csv', index_col=None, encoding='UTF-8', sep=';')
2  df_features_temp1 = df.iloc[:,0:10]
3  df_features_temp1['clientId_1'] = df_features_temp1['clientId_1'].str.replace('c','1')
4  df_features_temp1['clientId_1'] = pd.to_numeric(df_features_temp1.clientId_1, errors='coerce')
5  df_features_temp1['agentId_1'] = df_features_temp1['agentId_1'].str.replace('a','3')
6  df_features_temp1['agentId_1'] = df_features_temp1['agentId_1'].str.replace('css','399999')
7  df_features_temp1['agentId_1'] = pd.to_numeric(df_features_temp1.agentId_1, errors='coerce')
8  df_features_temp1['caseId_1'] = df_features_temp1['caseId_1'].str.replace('cid','5')
9  df_features_temp1['caseId_1'] = pd.to_numeric(df_features_temp1.caseId_1, errors='coerce').fillna(0).astype(np.int64)
10 df_features_temp1['contactDate_1'] = pd.to_datetime(df_features_temp1['contactDate_1'])
11 df_features_temp1['contactDate_1']= df_features_temp1['contactDate_1'].map(dt.datetime.toordinal)
12 df_onehotencoded_temp1 = pd.get_dummies(df_features_temp1)

13 df_features_temp2 = df.iloc[:,10:20]
14 df_features_temp2['clientId_2'] = df_features_temp2['clientId_2'].str.replace('c','1')
15 df_features_temp2['clientId_2'] = pd.to_numeric(df_features_temp2.clientId_2, errors='coerce')
16 df_features_temp2['agentId_2'] = df_features_temp2['agentId_2'].str.replace('a','3')
17 df_features_temp2['agentId_2'] = df_features_temp2['agentId_2'].str.replace('css','399999')
18 df_features_temp2['agentId_2'] = pd.to_numeric(df_features_temp2.agentId_2, errors='coerce')
19 df_features_temp2['caseId_2'] = df_features_temp2['caseId_2'].str.replace('cid','5')
20 df_features_temp2['caseId_2'] = pd.to_numeric(df_features_temp2.caseId_2, errors='coerce').fillna(0).astype(np.int64)
21 df_features_temp2['contactDate_2'] = pd.to_datetime(df_features_temp2['contactDate_2'])
22 df_features_temp2['contactDate_2']= df_features_temp2['contactDate_2'].map(dt.datetime.toordinal)
23 df_onehotencoded_temp2 = pd.get_dummies(df_features_temp2)

24 df_features_temp3 = df.iloc[:,20:30]
25 df_features_temp3['clientId_3'] = df_features_temp3['clientId_3'].str.replace('c','1')
26 df_features_temp3['clientId_3'] = pd.to_numeric(df_features_temp3.clientId_3, errors='coerce')
27 df_features_temp3['agentId_3'] = df_features_temp3['agentId_3'].str.replace('a','3')
28 df_features_temp3['agentId_3'] = df_features_temp3['agentId_3'].str.replace('css','399999')
29 df_features_temp3['agentId_3'] = pd.to_numeric(df_features_temp3.agentId_3, errors='coerce')
30 df_features_temp3['caseId_3'] = df_features_temp3['caseId_3'].str.replace('cid','5')
31 df_features_temp3['caseId_3'] = pd.to_numeric(df_features_temp3.caseId_3, errors='coerce').fillna(0).astype(np.int64)
32 df_features_temp3['contactDate_3'] = pd.to_datetime(df_features_temp3['contactDate_3'])
33 df_features_temp3['contactDate_3']= df_features_temp3['contactDate_3'].map(dt.datetime.toordinal)
34 df_onehotencoded_temp3 = pd.get_dummies(df_features_temp3)

35 df_features = pd.concat([df_onehotencoded_temp1, df_onehotencoded_temp2, df_onehotencoded_temp3], axis=1)
36 df_labels = df.iloc[:,-1] # last column of data frame with all rows
37 print("Features after one hot encoding::\n", list(df_features.columns), "\n")
38 print("Number of features after one hot encoding:\n", len(df_features.columns), "\n")
39 print("Shape of the data:\n", df_features.shape, "\n")
40 print("Shape of label:\n", df_labels.shape, "\n")
```

Listing 4.3: Preparing data for test run nr. 3

### Splitting the data

Using the *train_test_split()* function provided by *scikit-learn*, as can be seen in listing 4.4, the data which is in sequential order is separated into NumPy arrays containing the test and training data (Müller and Guido 2016, p.19). *X_train* contains 75% (536) of the vectors of the original data set and *X_test* contains the remaining 25% (159 vectors).

```
1 X_train, X_test, y_train, y_test = train_test_split(df_features,
      df_labels, random_state=0)
```

Listing 4.4: Splitting the data automatically into training and test data

For tests using the vectors which contain the unordered contact history entries, the *train_test_split()* function cannot be used, as this function first resorts the data in the DataFrame in a random order and only then splits it. In this case the data is manually split, as can be seen in listing 4.5. The 75% to 25% ratio is maintained in this manual split as well.

```
1 X_train = df_features.iloc[0:537]
2 X_test = df_features.iloc[537:715]

3 y_train = df_labels.iloc[0:537]
4 y_test = df_labels.iloc[537:715]
```

Listing 4.5: Splitting the data manually into training and test data

Further attempts were made to reduce the number of features to only the most useful ones, by using the strategies *univariate statistics*, *model-based selection* and *iterative selection*. (Müller and Guido 2016, p.238-243) As they did not lead to any change in performance, their results were omitted from the final test runs.

The complete source code written for each test can be found in Appendix C.

## 4.5. Training - logistic regression

*LogisticRegression* is a class provided by scikit-learn. Using the *fit()* function (see listing 4.6 on *line 2*) provided by the *LogisticRegression* class, the model is trained.

The function *predict()* on *line 3*, predicts the target values of *X_test* using the logistic regression model.

```
1 logreg = LogisticRegression()
2 logreg.fit(X_train, y_train)
3 pred_logreg = logreg.predict(X_test)
4 print(classification_report(y_test, pred_logreg))
```

Listing 4.6: Training implementation with logistic regression

### 4.5.1. Results and evaluation

scikit-learn provides a report for analyzing the precision, recall and f1-scores. As can be seen in listing 4.6 on *line 4*. The *classification_report()* function, generates an overview of the values required for the evaluation of the model. In this section this report will be used for the evaluation of each test run.

#### Test run nr. 1

Table 4.2 shows the results of the first test run. High precision relates to the low false positive rates, so an average value of 0.77 using an ordered sequence and an average of 0.80 for an unordered sequence is not bad. Similarly high recall relates to low false negative rates, so an average value of 0.77 or 0.80 is good in this case too. There is hardly any difference between the two trained models.

| | | precision | recall | f1-score |
|---|---|---|---|---|
| ordered vector | ... no complaint | 0.76 | 0.78 | 0.77 |
| | ... a complaint | 0.77 | 0.75 | 0.76 |
| | avg | 0.77 | 0.77 | 0.77 |
| unordered vector | ... no complaint | 0.82 | 0.79 | 0.80 |
| | ... a complaint | 0.79 | 0.82 | 0.80 |
| | avg | 0.80 | 0.80 | 0.80 |

Table 4.2.: Results of the first test run using logistic regression

### Test run nr. 2

Table 4.3 shows the results of the second test run. In comparison to the first run, all the evaluated values have become worse and the number of false positives and false negatives have increased. The precision and recall values are still over 0.70, but the aim is to improve the values with each run. Even with this run, there is almost no difference between the two models.

| | | precision | recall | f1-score |
|---|---|---|---|---|
| ordered vector | ... no complaint | 0.71 | 0.72 | 0.71 |
| | ... a complaint | 0.71 | 0.70 | 0.70 |
| | avg | 0.71 | 0.71 | 0.71 |
| unordered vector | ... no complaint | 0.73 | 0.69 | 0.71 |
| | ... a complaint | 0.70 | 0.74 | 0.72 |
| | avg | 0.71 | 0.71 | 0.71 |

Table 4.3.: Results of the second test run using logistic regression

### Test run nr. 3

Table 4.4 shows the results of the third test run. The changes made to the test and training data have paid off. Transforming the dates from strings to ordinals have had a great impact on the precision and recall values, leading to the best results of the three test runs using logistic regression. Even in this case, there is no real difference between the two trained models.

| | | precision | recall | f1-score |
|---|---|---|---|---|
| ordered vector | ... no complaint | 0.89 | 0.81 | 0.85 |
| | ... a complaint | 0.82 | 0.90 | 0.86 |
| | avg | 0.86 | 0.85 | 0.85 |
| unordered vector | ... no complaint | 0.88 | 0.86 | 0.87 |
| | ... a complaint | 0.85 | 0.87 | 0.86 |
| | avg | 0.87 | 0.87 | 0.87 |

Table 4.4.: Results of the third test run using logistic regression

## 4.6. Training - naive Bayes classifier

There are three kinds of naive Bayes classifiers implemented in scikit-learn: *BernoulliNB*, *MultinomialNB* and *GaussianNB*. While *BernoulliNB* assumes that the input is binary, *GaussianNB* assumes continuous data and *MultinomialNB* expects discrete (count) data. Like the LogisticRegression classifier, all the three naive Bayes classifiers offer the functions *fit()* and *predict()*, as can be seen in listing 4.7. These will be used to train and test the model. Each of the three naive Bayes classifiers will be tested and evaluated.

```
1  bernNB = BernoulliNB()
2  bernNB.fit(X_train, y_train)
3  pred_bernnb = bernNB.predict(X_test)
4  print(classification_report(y_test, pred_bernnb))

5  multiNB = MultinomialNB()
6  multiNB.fit(X_train, y_train)
7  pred_multinb = multiNB.predict(X_test)
8  print(classification_report(y_test, pred_multinb))

9  gausNB = GaussianNB()
10 gausNB.fit(X_train, y_train)
11 pred_gausnb = gausNB.predict(X_test)
12 print(classification_report(y_test, pred_gausnb))
```

Listing 4.7: Training implementation with naive Bayes

### 4.6.1. Results and evaluation

Using the report generated by the *classification_report()* function, all the test runs will be evaluated.

#### Test run nr. 1

Table 4.5 shows the results of the first test run. The Bernoulli and Multinomial classifiers performed similarly, where as the Gaussian classifier performed

slightly worse. Given the fact, that the input data completely consisted of binary values, it was expected that the Bernoulli classifier would perform the best. As was the case with linear regression, here too there is hardly any difference between the ordered and unordered model.

| | | | precision | recall | f1-score |
|---|---|---|---|---|---|
| BernoulliNB | ordered | ... no complaint | 0.89 | 0.68 | 0.78 |
| | | ... a complaint | 0.63 | 0.87 | 0.73 |
| | | avg | 0.79 | 0.75 | 0.76 |
| | unordered | ... no complaint | 0.83 | 0.76 | 0.79 |
| | | ... a complaint | 0.77 | 0.84 | 0.80 |
| | | avg | 0.80 | 0.80 | 0.80 |
| MultinomialNB | ordered | ... no complaint | 0.93 | 0.68 | 0.78 |
| | | ... a complaint | 0.63 | 0.91 | 0.75 |
| | | avg | 0.81 | 0.77 | 0.77 |
| | unordered | ... no complaint | 0.87 | 0.71 | 0.78 |
| | | ... a complaint | 0.75 | 0.89 | 0.81 |
| | | avg | 0.81 | 0.80 | 0.80 |
| GaussianNB | ordered | ... no complaint | 0.91 | 0.58 | 0.71 |
| | | ... a complaint | 0.57 | 0.91 | 0.70 |
| | | avg | 0.78 | 0.70 | 0.70 |
| | unordered | ... no complaint | 0.83 | 0.66 | 0.74 |
| | | ... a complaint | 0.71 | 0.86 | 0.78 |
| | | avg | 0.77 | 0.76 | 0.76 |

Table 4.5.: Results of the first test run using naive Bayes

#### Test run nr. 2

Table 4.6 shows the results of the second test run. While the Bernoulli classifier performs equally well as the first run, the Multinomial classifier worsens signifi-

cantly. The Gaussian classifier performs by far the best.

As the test data for the second run no more contains the *clientId*, *agentId* and *caseId* as binary values, it was expected that the Bernoulli classifier would perform significantly worse, given the fact that the Bernoulli classifier would transform all non binary values into binary value using the threshold of 0.0. This however was not the case. As expected, the Gaussian classifier performed the best, given the type of the input data.

| | | | precision | recall | f1-score |
|---|---|---|---|---|---|
| BernoulliNB | ordered | ... no complaint | 0.88 | 0.68 | 0.77 |
| | | ... a complaint | 0.62 | 0.85 | 0.72 |
| | | avg | 0.78 | 0.74 | 0.75 |
| | unordered | ... no complaint | 0.87 | 0.65 | 0.74 |
| | | ... a complaint | 0.71 | 0.90 | 0.79 |
| | | avg | 0.79 | 0.77 | 0.77 |
| MultinomialNB | ordered | ... no complaint | 0.76 | 0.67 | 0.71 |
| | | ... a complaint | 0.54 | 0.65 | 0.59 |
| | | avg | 0.67 | 0.66 | 0.66 |
| | unordered | ... no complaint | 0.77 | 0.66 | 0.71 |
| | | ... a complaint | 0.69 | 0.79 | 0.74 |
| | | avg | 0.73 | 0.72 | 0.72 |
| GaussianNB | ordered | ... no complaint | 0.98 | 0.82 | 0.89 |
| | | ... a complaint | 0.77 | 0.97 | 0.86 |
| | | avg | 0.90 | 0.88 | 0.88 |
| | unordered | ... no complaint | 0.97 | 0.85 | 0.91 |
| | | ... a complaint | 0.86 | 0.98 | 0.91 |
| | | avg | 0.92 | 0.91 | 0.91 |

Table 4.6.: Results of the second test run using naive Bayes

### Test run nr. 3

Table 4.7 shows the results of the third test run. It is expected that the Gaussian classifier performs the best, as the input data was further cleaned for the third test run. It was also expected that the Bernoulli classifier would perform worse, as the input data now contains the *contactDate* in ordinal instead of a binary form. These expectations were met. However the Gaussian classifier did not perform much better that the previous run even though the number of false positives were significantly reduced. Similar to the test runs with the logistic regression model, there are no big differences between the ordered and unordered model.

| | | | precision | recall | f1-score |
|---|---|---|---|---|---|
| BernoulliNB | ordered | ... no complaint | 0.84 | 0.65 | 0.73 |
| | | ... a complaint | 0.58 | 0.79 | 0.67 |
| | | avg | 0.74 | 0.70 | 0.71 |
| | unordered | ... no complaint | 0.84 | 0.63 | 0.72 |
| | | ... a complaint | 0.69 | 0.87 | 0.77 |
| | | avg | 0.77 | 0.75 | 0.74 |
| MultinomialNB | ordered | ... no complaint | 0.75 | 0.66 | 0.70 |
| | | ... a complaint | 0.54 | 0.65 | 0.59 |
| | | avg | 0.67 | 0.65 | 0.66 |
| | unordered | ... no complaint | 0.77 | 0.66 | 0.71 |
| | | ... a complaint | 0.69 | 0.79 | 0.74 |
| | | avg | 0.73 | 0.72 | 0.72 |
| GaussianNB | ordered | ... no complaint | 1.00 | 0.78 | 0.88 |
| | | ... a complaint | 0.74 | 1.00 | 0.85 |
| | | avg | 0.90 | 0.87 | 0.87 |
| | unordered | ... no complaint | 0.99 | 0.86 | 0.92 |
| | | ... a complaint | 0.87 | 0.99 | 0.92 |
| | | avg | 0.93 | 0.92 | 0.92 |

Table 4.7.: Results of the third test run using naive Bayes

## 4.7. Training - support vector machine

Along with linear regression, the linear support vector machine is the most common linear classification algorithm. Like the previous algorithms, *LinearSVC* is also provided by scikit-learn. Like the LogisticRegression classifier, this classifier also offers the functions *fit()* and *predict()*, as can be seen in listing 4.8.

The LinearSVC implements the "one-vs-the-rest" multi-class strategy, thereby training n_class models. If there are only two classes, as is the case in this test, only one model is trained.

```
1 svmAlg = LinearSVC(random_state=0)
2 svmAlg.fit(X_train, y_train)
3 pred_svm = svmAlg.predict(X_test)
4 print(classification_report(y_test, pred_svm))
```

Listing 4.8: Training implementation with SVM

### 4.7.1. Results and evaluation

Using the report generated by the *classification_report()* function, all the test runs will be evaluated.

#### Test run nr. 1

Table 4.8 shows the results of the first test run. Both the model with the ordered sequence, as well as the model with the unordered sequence perform the same. Similar to the first logistic regression test run, this model generates high precision and recall values.

| | | precision | recall | f1-score |
|---|---|---|---|---|
| ordered vector | ... no complaint | 0.79 | 0.80 | 0.80 |
| | ... a complaint | 0.80 | 0.79 | 0.79 |
| | avg | 0.79 | 0.79 | 0.79 |
| unordered vector | ... no complaint | 0.80 | 0.75 | 0.77 |
| | ... a complaint | 0.75 | 0.80 | 0.78 |
| | avg | 0.78 | 0.78 | 0.78 |

Table 4.8.: Results of the first test run using the linear svm

#### Test run nr. 2

Table 4.9 shows the results of the second test run. Both the models perform badly using the given input data, resulting in average precision and recall values below 0.5. Surprisingly the model with the unordered sequence performs significantly better, but the precision and recall values show that this model is not useable and will provide lots of false negatives and hardly any true positives.

| | | precision | recall | f1-score |
|---|---|---|---|---|
| ordered vector | ... no complaint | 0.32 | 0.23 | 0.27 |
| | ... a complaint | 0.39 | 0.51 | 0.44 |
| | avg | 0.36 | 0.37 | 0.36 |
| unordered vector | ... no complaint | 0.51 | 1.00 | 0.68 |
| | ... a complaint | 1.00 | 0.01 | 0.02 |
| | avg | 0.75 | 0.52 | 0.36 |

Table 4.9.: Results of the second test run using the linear svm

#### Test run nr. 3

Table 4.10 shows the results of the third test run. Both the models don't perform generally well. While the model using the ordered vectors returns many false

positives, the other model returns lots of false negatives and hardly any true positives. This time however the model with the unordered vectors performs worse than the model with the ordered vectors. Still both these models are not really useable for the given classification problem.

| | | precision | recall | f1-score |
|---|---|---|---|---|
| ordered vector | ... no complaint | 0.94 | 0.53 | 0.68 |
| | ... a complaint | 0.67 | 0.97 | 0.79 |
| | avg | 0.81 | 0.75 | 0.74 |
| unordered vector | ... no complaint | 0.51 | 1.00 | 0.68 |
| | ... a complaint | 1.00 | 0.01 | 0.02 |
| | avg | 0.75 | 0.52 | 0.36 |

Table 4.10.: Results of the third test run using the linear svm

## 4.8. Training - decision trees

Decision trees learn a hierarchy of if/else questions leading to a decision. In order to combat overfitting for decision trees, there are two common strategies that can be implemented: stopping the creation of the tree early (*pre-pruning*) or building the tree but then removing nodes containing little information (*post-pruning*). As scikit-learn only implements *pre-pruning*, that will be the strategy chosen. *Pre-pruning* can be implemented by limiting the maximum depth of the tree. For all the test runs, the maximum depth will be set to four, as can be seen in listing 4.9 on line 1. This limit was chosen, as it lead to the best results. Like all the previous classifier, this classifier also offers the functions *fit()* and *predict()*, as can be seen in listing 4.9.

```
1 decisionTree = tree.DecisionTreeClassifier(random_state=0,
      max_depth=4)
2 decisionTree.fit(X_train, y_train)
3 pred_decTree = decisionTree.predict(X_test)
4 print(classification_report(y_test, pred_decTree))
```

Listing 4.9: Training implementation with decision trees

### 4.8.1. Results and evaluation

Using the report generated by the *classification_report()* function, all the test runs will be evaluated.

#### Test run nr. 1

Table 4.11 shows the results of the first test run. Surprisingly the model with the unordered vectors performs better that then model with the ordered vectors. They are both not very useable as they predict too many false positives and false negatives.

| | | precision | recall | f1-score |
|---|---|---|---|---|
| ordered vector | ... no complaint | 0.72 | 0.62 | 0.67 |
| | ... a complaint | 0.66 | 0.75 | 0.71 |
| | avg | 0.69 | 0.69 | 0.69 |
| unordered vector | ... no complaint | 0.88 | 0.57 | 0.69 |
| | ... a complaint | 0.67 | 0.92 | 0.78 |
| | avg | 0.78 | 0.74 | 0.73 |

Table 4.11.: Results of the first test run using the decision tree

#### Test run nr. 2

Table 4.12 shows the results of the second test run. Using the data set prepared for the second run, as explained in section 4.4, there is a significant improvement

in the precision and recall values. Both the model with the ordered vectors as well as the model with the unordered vectors offer similarly high precision and recall values. Both models don't return many false positives and false negatives, making them quite accurate.

| | | precision | recall | f1-score |
|---|---|---|---|---|
| ordered vector | ... no complaint | 0.93 | 0.84 | 0.88 |
| | ... a complaint | 0.86 | 0.93 | 0.89 |
| | avg | 0.89 | 0.89 | 0.89 |
| unordered vector | ... no complaint | 0.97 | 0.85 | 0.91 |
| | ... a complaint | 0.86 | 0.98 | 0.91 |
| | avg | 0.92 | 0.91 | 0.91 |

Table 4.12.: Results of the second test run using the decision tree

#### Test run nr. 3

Table 4.13 shows the results of the third test run. The third test run where the *contactDate* is now treated as an ordinal value, offers slight improvement. Both models offer similarly high precision and recall values. Just like the second test run, in this test run not many false positive and false negatives are returned, making the models quite accurate.

| | | precision | recall | f1-score |
|---|---|---|---|---|
| ordered vector | ... no complaint | 1.00 | 0.79 | 0.88 |
| | ... a complaint | 0.82 | 1.00 | 0.90 |
| | avg | 0.91 | 0.89 | 0.89 |
| unordered vector | ... no complaint | 0.92 | 0.92 | 0.92 |
| | ... a complaint | 0.92 | 0.92 | 0.92 |
| | avg | 0.92 | 0.92 | 0.92 |

Table 4.13.: Results of the third test run using the decision tree

## 4.9. Comparison

The results of all the tests show that even with a short sequence of contact history entries, it is possible to train a model that can classify a sequence of contact history entries as *leading to a complaint* or as *not leading to a complaint* quite accurately.

The main research question in section 1.3 questioned if it would be possible to better identify potential customer complaints in advance if the interactions are explicitly described in sequences instead of a disorderly collection of events. Based on the results in table 4.14 the answer should be no, as for all algorithms both the models performed similarly. However it is the author's opinion that using vectors with significantly more than three entries could potentially show that learning with sequences is better than learning with a disorderly collection of events.

As can be seen in table 4.14, three of the algorithms performed really well. Only the support vector machine couldn't achieve similar precision and recall values as the others. Based on the results, the choice is between the naive Bayes and the decision tree model. As they both perform equally well, making a choice between these two will need to be based on other criteria, such as understandability, simplicity in usage, computational requirement, etc.

| algorithm | | precision | recall | f1-score |
|---|---|---|---|---|
| linear regression | ordered vector | 0.86 | 0.85 | 0.85 |
| | unordered vector | 0.87 | 0.87 | 0.87 |
| naive Bayes | ordered vector | 0.90 | 0.87 | 0.87 |
| | unordered vector | 0.93 | 0.92 | 0.92 |
| svm | ordered vector | 0.79 | 0.79 | 0.79 |
| | unordered vector | 0.78 | 0.78 | 0.78 |
| decision trees | ordered vector | 0.91 | 0.89 | 0.89 |
| | unordered vector | 0.92 | 0.92 | 0.92 |

Table 4.14.: Overview of the best average results per algorithm

# 5. System design concept

## 5.1. Introduction

In this section a system will be conceptually designed and described that can predict potential customer complaints from existing customer interactions in advance, thus answering one of the research questions posed in section 1.3.

The system will be designed using requirements acquired from the expert interview conducted with Fredi Bircher (see Appendix A.3). Further input for the system in design will be taken from the experiment results in section 4.9. As mentioned in section 1.4, this new complaint prediction system will be integrated into the current IT architecture of CSS. The system in design will be discussed with Fredi Bircher to ensure compliance with the requirements.

## 5.2. Current situation

### 5.2.1. IT landscape

The CSS IT landscape is composed of multiple core business applications that communicate with each other. Some of the core applications are the CRM system, the Data warehouse (DWH) system, the document management system (DMS) and the claims system. CSS uses a SOA architecture model on which it operates its distributed and heterogeneous IT landscape in an agile and cost-effective way.

The core business applications communicate with each other synchronously using business services and asynchronously using events (see figure 5.1). This avoids redundancies of logic and data and enables reusability.

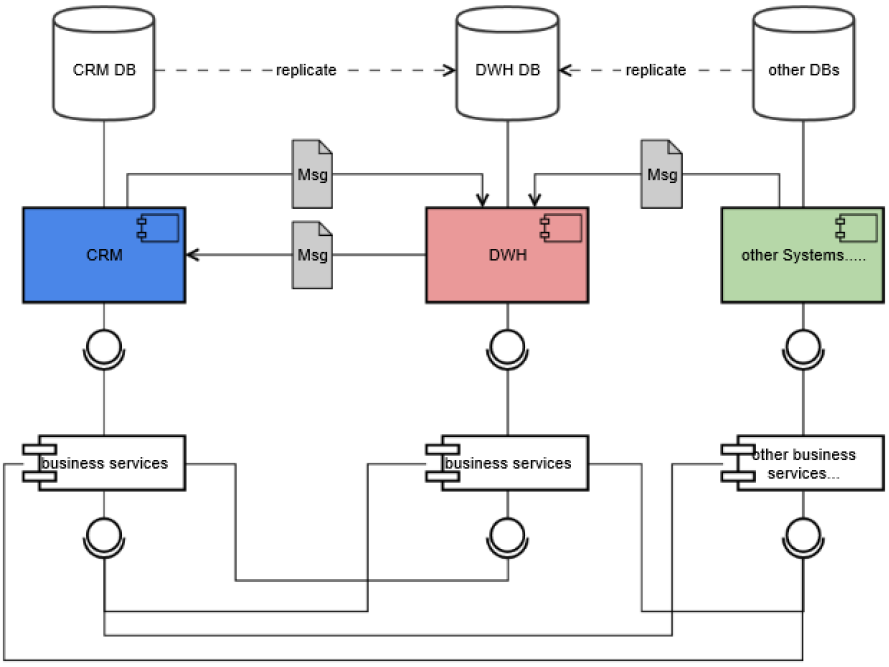


Figure 5.1.: Rough overview of the CSS IT landscape

### 5.2.2. Information flow

The core system the insurance advisors use in order to collect customer information or to mutate customer data is the CRM system. This is the system they use daily and where they maintain their list of pending tasks. The tasks can come in different forms, such as customer concerns, sales opportunities for new customers, service requests or leads.

There are a variety of different leads, depending on their type (back-office leads, retention leads, info leads, campaigns, etc.) different people with different roles are responsible for handling them. These leads are not generated in the CRM system, as this system does not contain the analytical knowhow or functionality required for that. The system responsible for generating the leads is the DWH system.

The DWH system aggregates data from all the core systems using the business services these systems provide, using database replication and sometimes by consuming events and storing the data in their database. A rough overview of this

can be seen in figure 5.1. For example, when a customer turns eighteen years old, an info lead is generated by DWH and sent asynchronously via an event to the CRM system and assigned to a certain employee. The employee can then see the info lead in the CRM system in their list of pending tasks and can then call the customer and complete the task they are given.

## 5.3. Requirements

CSS aims to increase its customer satisfaction ratings and become number one in Switzerland. Handling complaints better, plays a central role in achieving this goal. There are two strategies to go about handling customer complaints: being reactive and being proactive. Depending on the choice of strategy, the consequences are different, as can be seen in figure 5.2.

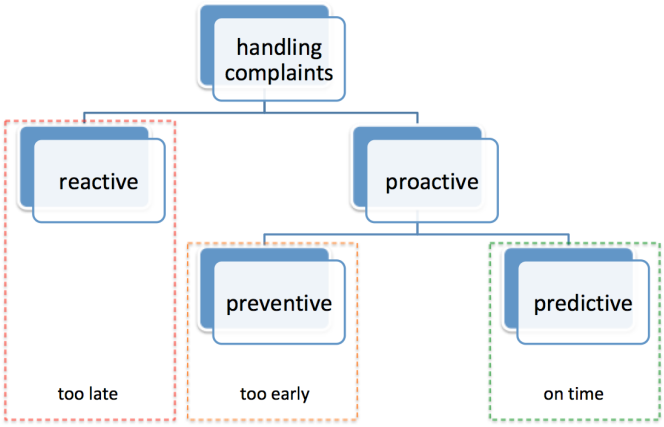


Figure 5.2.: Difference between reactive and proactive strategies (based on *The difference between preventive and predictive maintenance* 2018)

Based on the expert interview conducted with Fredi Bircher (see Appendix A.3), following three high level requirements were specified:

1. It is important that insurance advisors not wait for the customer to call and complain *(reactive)*, but be smart and contact the customer in advance if needed *(predictive)*.

2. As there are multiple events that take place periodically (e.g. dunning process) which could potentially lead to complaints, the complaint prediction should be done periodically as well.

3. The process of informing the insurance advisors should use the existing infrastructure and mechanisms.

## 5.4. Design concept

Based on the high level requirements in section 5.3, following design decisions were made:

- In order to fulfil requirements nr. 1, a model should be learnt using the data from the customer contact history. As evaluated in section 4.9, either the naive Bayes or the decision tree algorithm should be used to train the model, as these have performed the best.

- Given the analytical and predictive nature of this task, the trained model and supporting services should be provided by DWH.

- As the CRM system is the main system the insurance advisors use, they should receive information about potential complaints in this system.

- The information about a potential complaint should be given to them in form of a new lead type, the **complaint lead**.

- This new lead should be generated by the DWH system and then sent asynchronously to the CRM system. This will fulfil requirements nr. 3.

- In order to generate these new leads periodically, as specified in requirements nr. 2, a batch program should be written that can be configured to run whenever required. This batch program will asynchronously send the leads to the CRM system on the given day.

As DWH is the designated provider of this predictive service, the question arises how they can aggregate the contact history data. Like the CRM system,

they would need to use the business services provided by the relevant systems to collect and persist the data. These services always provide all the historical data of the last six months, which would mean, that the DWH system would need to filter out the newest records for each customer and only append those to the customer specific contact history list they persist. This would be highly inefficient, as they would have to collect the data from all the customers in order to update their historical customer contact history database.

A better solution would be to initially load the relevant data via database replication into the DWH system. Subsequently the other core systems could inform the DWH system proactively when a new contact history record is generated in their system or when an update is made to a current one. This can be done asynchronously with the help of events. As the DWH system already subscribes to different events and consumes them, subscribing to these new events would not lead to big changes in the DWH system. It would however require that all the other core systems publish these new events on creation or update of a contact history entry. Should there be a need to enrich the data, the DWH system can use the business services provided by the other systems. Figure 5.3 gives a high level overview of the proposed process.

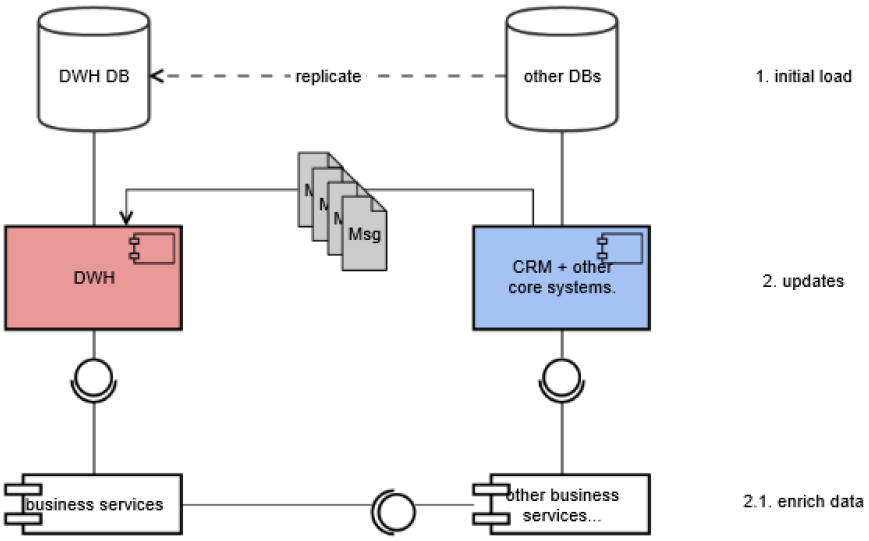


Figure 5.3.: Overview of the new data aggregation process for DWH

# 6. Summary and conclusion

The main objective of this research was to find a way to predict possible customer complaints in advance using the customer contact history generated by CSS. It was also questioned if treating the contact history as an ordered sequence was better for prediction than treating it as an unordered collection of events. In this thesis, the feasibility of a complaint prediction model is shown theoretically and empirically.

Even after limiting the length of the sequence vector to three contact history entries, after excluding multiple features from the data set for data protection reasons and after limiting the choice of algorithms for computational reasons, good precision and recall values could be achieved. By further manipulating the data using the process of one hot encoding and ordinal conversion, further improvements to the precision and recall values were achieved. No significant difference could be found in the performance of the models using vectors with an ordered sequence of contact history entries and vectors without an ordered sequence. It is shown that the naive Bayes or the decision trees algorithms is the one to select when trying to solve this research problem.

In the opinion of the author the trained models could be further improved by using longer sequences for the input vectors and by drilling down further into the data sets and enriching them with more information from the relevant systems. Identifying what information more to add will be a big challenge, but involving experts from the complaints management department will help to narrow down the choice. The author believes that using longer input vectors will show that using vectors with an ordered sequence of entries will lead to better and more accurate results than using vectors without an ordered sequence. Also the need to use more complex but less explainable models such as neural networks or markov models is not given, as it was shown that the naive Bayes and the decision trees

algorithm could perform well with the given data.

Furthermore, it was analysed how to automatically portray and convert customer contact history entries in SML. The fact that each entry contains multiple enumerations, leads to a large amount of permutations in the narrative. This complicated the automation process significantly.

Finally a complaint prediction system was conceptually described in the context of CSS. Using the high level requirements specified by Fredi Bircher, the PO of the scrum team "customer services", a possible solution was described. It was proposed that the responsibility of aggregating the data, learning the model and providing the relevant services, should be provided by the DWH system. In order to use the existing infrastructure and processes, it was proposed that the insurance advisors would be periodically informed about potential complaints using a new type of lead (*complaint lead*) in the CRM system.

In order to eliminate pointless business service calls and to enable a higher frequency for complaint prediction in the future, it was proposed that the DWH system should be proactively informed about changes or additions made to a customer's contact history. This can be accomplished using asynchronous communication with events. Always being up-to-date with the customer contact history entries will not only enable the DWH system to potentially generate new predictions as soon as the data changes, but also to retrain the prediction model at any time.

In closing it can be said that the main objective of this research was met. Given more time and resources, the author would have liked to implement a productive prediction system, which could have been used as a proof of concept. This could have further interested the business in investing in machine learning for complaint prediction.

# 7. Outlook

The research work presented in this thesis is a first attempt at predicting potential complaints using customer interaction data. It can be used as a good starting off point for further research in this area. It would be interesting to see if using the more complex and powerful neural network would produce significantly better result. Another interesting approach would be to investigate, optimize and enrich the data further. This could be done by involving experts from the relevant business areas. With an optimal setting for the naive Bayes and decision tree classifier, one could probably achieve even better precision and recall values.

The idea of a layman with no IT background being able to analyse data in the form of stories using SML queries, was looked upon as very interesting by CSS. However, at the moment there is no project planned in the foreseeable future and no concrete use case defined, where SML would provide a solution.

For CSS the next step would be to implement a system according to the design concept specified in section 5.4 and to roll it out in the production environment for a pilot group. From a business point of view it would be interesting to see how the insurance advisors react to this new *complaint lead*, but more importantly how being proactive and predictive help CSS improve their customer satisfaction ratings. On the technical side it would be a good challenge to find out how to setup such a system in a productive environment and how to handle the ML model life cycle. This is a challenge the author will gladly take up with the help and resources provided by CSS.

# References


Abts, Dietmar and Wilhelm Mülder (2017). *Grundkurs Wirtschaftsinformatik - Eine kompakte und praxisorientierte Einführung*. 9., erweiterte und aktualisierte Auflage. Mönchengladbach: Springer Vieweg, Wiesbaden. ISBN: 978-3-658-16378-5. DOI: 10.1007/978-3-658-16379-2_14. URL: https://link.springer.com/chapter/10.1007/978-3-658-16379-2_14 (visited on 11/12/2017).

Bari, Anasse, Mohamed Chaouchi, and Tommy Jung (2016). *Predictive Analytics für Dummies*. 1. Edition. Wiley.

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 1. Edition. 978-0387310732. Springer.

Brown, Iain and Christophe Mues (Feb. 15, 2012). "An experimental comparison of classification algorithms for imbalanced credit scoring data sets". In: *Expert Systems with Applications* 39.3, pp. 3446–3453. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2011.09.033. URL: http://www.sciencedirect.com/science/article/pii/S095741741101342X.

Brush, John (May 22, 2017). *Story Modeller*. Winterthur. URL: https://www.youtube.com/watch?v=ceaIb03uUNA&feature=youtu.be (visited on 10/17/2017).

Comparis (2017). *Schweizer Krankenkassen*. URL: https://www.comparis.ch/krankenkassen/umfrage/hitlist (visited on 05/10/2018).

CSS Krankenversicherung AG (2015). *Unveröffentlichtes Dokument: CSS Vision 2018*.

– (Oct. 11, 2017). *CSS Gruppe*. CSS Gruppe. URL: https://www.css.ch/de/home/ueber_uns/unternehmen/gruppengesellschaft.html (visited on 11/10/2017).

Dante (Mar. 17, 2017). *Kernel trick explanation*. Data Science Stack Exchange. URL: https://datascience.stackexchange.com/questions/17536/kernel-trick-explanation (visited on 05/10/2018).

Deloitte Consulting (2015). *Disruption ahead - Deloitte's point of view on IBM Watson*. URL: https://www2.deloitte.com/content/dam/Deloitte/us/Documents/about-deloitte/us-ibm-watson-client.pdf (visited on 11/11/2017).

Dietterich, Thomas G. (Aug. 2002). "Machine Learning for Sequential Data: A Review". In: *Structural, Syntactic, and Statistical Pattern Recognition*. Springer, Berlin, Heidelberg, pp. 15–30.

Duda, Richard O., Peter E. Hart, and David G. Stork (Nov. 2000). *Pattern Classification*. 2. Edition. Wiley-Interscience.

Fettke, Peter (Aug. 1, 2006). "State of the Art of the State of the Art — A study of the research method „Review" in the information systems discipline". In: *WIRTSCHAFTSINFORMATIK* 48.4, p. 257. ISSN: 0937-6429, 1861-8936. DOI: 10.1007/s11576-006-0057-3. URL: https://link.springer.com/article/10.1007/s11576-006-0057-3 (visited on 11/09/2017).

Gartner Inc. (Aug. 16, 2012). *Gartner's 2012 Hype Cycle for Emerging Technologies Identifies "Tipping Point" Technologies That Will Unlock Long-Awaited Technology Scenarios*. URL: https://www.gartner.com/newsroom/id/2124315 (visited on 11/11/2017).

– (Aug. 16, 2016). *Gartner's 2016 Hype Cycle for Emerging Technologies Identifies Three Key Trends That Organizations Must Track to Gain Competitive Advantage*. URL: https://www.gartner.com/newsroom/id/3412017 (visited on 11/11/2017).

– (Aug. 15, 2017). *Top Trends in the Gartner Hype Cycle for Emerging Technologies, 2017*. Smarter With Gartner. URL: https://blogs.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/ (visited on 11/09/2017).

Graves, Alex (Jan. 2012). *Supervised Sequence Labelling with Recurrent Neural Networks*. Vol. 385. 978-3642247965. Springer.

Hevner, Alan R. et al. (Mar. 2004). "Design Science in Information Systems Research". In: *MIS Q.* 28.1, pp. 75–105. ISSN: 0276-7783. URL: http://dl.acm.org/citation.cfm?id=2017212.2017217 (visited on 11/09/2017).

Hong, Liangjie, Ovidiu Dan, and Brian D. Davison (2011). "Predicting popular messages in Twitter." In: *WWW (Companion Volume)*. Ed. by Sadagopan Srinivasan et al. ACM, pp. 57–58. ISBN: 978-1-4503-0637-9. URL: http://dblp.uni-trier.de/db/conf/www/www2011c.html#HongDD11.

Huang, Xiaohong and Wei Pan (2003). "Linear regression and two-class classification with gene expression data". In: *Bioinformatics* 19.16, pp. 2072–2078. DOI: 10.1093/bioinformatics/btg283. eprint: /oup/backfile/content_public/journal/bioinformatics/19/16/10.1093/bioinformatics/btg283/2/btg283.pdf. URL: http://dx.doi.org/10.1093/bioinformatics/btg283.

Lamnek, Siegfried and Claudia Krell (Nov. 7, 2016). *Qualitative Sozialforschung - Mit Online-Materialien - Siegfried Lamnek, Claudia Krell | BELTZ*. 6. Auflage. ISBN: 978-3-621-28269-7. URL: https://www.beltz.de/fachmedien/psychologie/buecher/produkt_produktdetails/29398-qualitative_sozialforschung.html (visited on 11/12/2017).

Maurer, Jürgen (Aug. 26, 2015). *Big-Data-Trends im Überblick: Was ist was bei Predictive Analytics?* Computerwoche. URL: https://www.computerwoche.de/a/was-ist-was-bei-predictive-analytics,3098583 (visited on 11/11/2017).

Meuser, Michael and Ulrike Nagel (2002). "ExpertInneninterviews — vielfach erprobt, wenig bedacht". In: *Das Experteninterview*. VS Verlag für Sozialwissenschaften, Wiesbaden, pp. 71–93. ISBN: 978-3-322-93270-9. DOI: 10.1007/978-3-322-93270-9_3. URL: https://link.springer.com/chapter/10.1007/978-3-322-93270-9_3 (visited on 11/12/2017).

Müller, Andreas C. and Sarah Guido (2016). *Introduction to Machine Learning with Python*. 1. Edition. O'Reilly.

Netzer, Oded, James Lattin, and V Srinivasan (June 1, 2007). "A Hidden Markov Model of Customer Relationship Dynamics". In: *Marketing Science* 27. DOI: 10.2139/ssrn.776765.

Pustejovsky, James and Amber Stubbs (2012). *Natural Language Annotation for Machine Learning*. 1. Edition. O'Reilly. ISBN: 978-1-4493-0666-3.

Ren, X. and J. Malik (Oct. 2003). "Learning a classification model for segmentation". In: *Proceedings Ninth IEEE International Conference on Computer Vision*, 10–17 vol.1. DOI: 10.1109/ICCV.2003.1238308.

Spindler, Alexandre de and Max Meisterhans (2017). "Powerpoint-Präsentation: Story Modeller Prototype - Capture and Query Stories". Winterthur: Zürcher Hochschule für angewandte Wissenschaften.

Sutherland, Jeff and Ken Schwaber (Nov. 2017). *The Scrum Guide*. URL: http://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100 (visited on 11/12/2017).

Swisscom (Schweiz) AG (June 2014). *Predictive Analytics – Die Königsdisziplin im Analytischen CRM*. URL: http://www.swisscom.com/its/dam/documents/factsheets/fs_predictive_analytics_de.pdf (visited on 11/11/2017).

Team, Azure Machine Learning (Dec. 2017). *Microsoft Azure Machine Learning Algorithm Cheat Sheet*. Microsoft Corporation.

*The difference between preventive and predictive maintenance* (2018). URL: https://www.exsyn.com/blog/big-data-in-aviation-predictive-maintenance (visited on 05/19/2018).

Xu, Qianli et al. (2009). "An analytical Kano model for customer need analysis". In: *Design Studies* 30.1, pp. 87–110. ISSN: 0142-694X. DOI: 10.1016/j.destud.2008.07.001. URL: http://www.sciencedirect.com/science/article/pii/S0142694X0800063X (visited on 11/11/2017).

# A. Interviews

## A.1. Interview preparation and introduction

### 1 Vorbereitung und Einführung

| | |
|---|---|
| Datum: | [Datum] |
| Ort: | CSS Versicherung AG, Tribschenstrasse 21, 6005 Luzern |
| Interviewpartner: | Herr Fredi Bircher, Product Owner |
| Interviewer: | Rahul Rao |

### 1.1 Plan

| Themenbereich | Beschreibung | Dauer |
|---|---|---|
| **Begrüssung** | Zuerst wird der Interviewpartner begrüsst und für seine Teilnahme bedankt. Es wird erläutert, dass das Interview ca. zwanzig Minuten dauern wird. | 1’ |
| **Scope** | Das Projekt wird zuerst vorgestellt. Anschliessend wird erklärt, dass aus dieser Expertenbefragung heraus Anforderungen an den Prototypen spezifiziert werden. Zusätzlich soll der IST-Prozess geklärt und Unklarheiten ausgeräumt werden. Es sollen gegenüber dem Experten die Abgrenzungen kommuniziert werden. | 5’ |
| **Vertraulichkeit** | Das Interview wird aufgezeichnet, damit es anschliessend sinngemäss transkribiert und ausgewertet werden kann. Die Angaben des Interviews werden vertraulich behandelt und ausschliesslich für die Projektarbeit verwendet. Falls der Interviewpartner dies nicht wünscht, werden Notizen vom Autor gemacht. | 1’ |
| **Fragen?** | Bevor die Aufnahme des Gesprächs startet, werden allfällige Unklarheiten und offene Fragen geklärt. | 1’ |
| **Interview** | Das Interview wird anhand des Leitfadens durchgeführt. | 10’ |
| **Abschluss** | Zum Abschluss werden die nächsten Schritte besprochen und wenn nötig Vereinbarungen getroffen. Der Interviewpartner wird für seine Teilnahme bedankt. | 3’ |

# A.2. Interview guide for the requirements

## 2 Interview Leitfaden – Anforderungen

| Themenbereich | Fragen |
|---|---|
| **Warm-Up** | • Bitte beschreibe kurz deine Rolle im Unternehmen.<br>• Was verstehst du unter Machine Learning und hast du bereits Erfahrung damit? |
| **Prozess** | • Wie geht der Serviceline Mitarbeitende typischerweise vor wenn ein Anruf von einem Kunden reinkommt?<br>• Wann soll der Mitarbeitende im Prozess über die Wahrscheinlichkeit einer Beschwerde informiert werden?<br>• Was macht der Mitarbeiter mit dieser Information? |
| **Technisch / Daten** | • Soll das Machine Learning Modell im produktiven System laufend oder nur periodisch dazulernen?<br>• Gibt es deiner Meinung nach Informationen in der Kontakthistorie die keinen Einfluss darauf haben, ob das nächste Ereignis eine Beschwerde sein wird?<br>• Gibt es deiner Meinung nach Informationen in der Kontakthistorie, die einen Einfluss darauf haben, ob das nächste Ereignis eine Beschwerde sein wird? |

## A.3. Interview - Fredi Bircher

**Interviewer:** Rahul Rao (RR)
**Interview partner:** Fredi Bircher (FB)

RR: Bitte beschreibe kurz deine Rolle im Unternehmen.

FB: Ich bin Product Owner für das Produkt Kundenbetreuung. Das beinhaltet primär den Prozess bei dem der Kunde mit uns in Kontakt tritt und wir ihn möglichst optimal unterstützen.

RR: Du bist für diesen Prozess über alle Kanäle hinweg verantwortlich?

FB: Ja genau, über alle Kanäle hinweg. Ziel des Produktes, beziehungsweise unsere Vision ist die Kunden auf all den Kanälen optimal zu unterstützen und ihre Anliegen zu bearbeiten.

RR: Was verstehst du unter Machine Learning und hast du bereits Erfahrungen damit? Habt ihr bereits etwas in diesem Bereich umgesetzt oder habt ihr etwas vor?

FB: Erfahrungen habe ich nicht. Wir haben einfach Mal diskutiert, dass es gut wäre wenn man prädiktiv Anliegen erkennen könnte, welche der Kunde hat, damit man entsprechend darauf reagieren oder man ihn besser bedienen könnte. Oder sogar vorgängig darauf agieren. Also dieses Vorhaben existiert nur als Idee und nicht mehr.

RR: Nun würde ich gerne Fragen zum Prozess, zum IST–Zustand stellen. Wie geht der Serviceline Mitarbeitende vor wenn ein Anruf von einem Kunden reinkommt?

FB: Der erste Schritt ist bereits die Identifikation des Kunden. Wenn das System die Telefonnummer des Kunden erkennt, passiert das automatisch. Der Serviceline Mitarbeitende muss auch in diesem Fall überprüfen ob die automatische Erkennung stimmt. Meistens kann er dies mittels zwei bis drei Fragen über Name, Geburts–datum oder Wohnort klären. Dann muss der Mitarbeitende zuhören und das Anliegen vom Kunden erkennen. Nachher muss er versuchen das Anliegen so gut wie möglich zu beantworten und anschliessend muss er das natürlich dokumentieren. Für das braucht er eine möglichst gute Übersicht über den Kunden. Er muss sehr schnell zu den notwendigen Informationen kommen, die er für das Kundenanliegen benötigt. Auch die Dokumentation muss möglichst effizient passieren.

RR: Das heisst er versucht so gut wie möglich alles selber zu lösen?

FB: Also das Ziel welches wir uns setzen ist, dass wir möglichst alle Anliegen mit dem Erstkontakt abschliessen können. Es soll möglichst selten ein Experte konsultiert werden und das Anliegen sollte auch selten an ein Second– oder Third–Level weitergereicht werden.

RR: Gibt es heute schon eine bestimmte Vorgehensweise um mit Kunden umzugehen die sich beschweren?

FB: Ich bin in diesem Bereich zu wenig fachlich drin. Aber ich weiss, dass es viele Schulungen gibt bezüglich Verhaltensregeln um die Mitarbeiter darauf zu trainieren wie man mit unterschiedlichen Kunden–situationen umgehen sollte. Wenn zum Beispiel ein Kunde aggressiv reagiert, wie soll man damit umgehen.

RR: Das heisst es wäre hilfreich, wenn diese Mitarbeitende bereits am Anfang des Gespräches wüssten mit welcher Wahrscheinlichkeit der Kunde sich beschweren möchte. Sie könnten dann mit dem richtigen Mindset das Gespräch beginnen.

FB: Es ist schwierig zu sagen ob das gut wäre diese Informationen direkt vor dem Gespräch zu erhalten. Denn wenn man mit dieser Einstellung in das Gespräch hineingeht, kann es sein, dass man nicht mehr offen ist. Es kann sein, dass du nicht mehr neutral auf den Kunden reagierst. Man müsste das ausprobieren. Ich kann jetzt nicht sagen ob diese Informationen das Gespräch wirklich positiv beeinflussen würde oder ob das eher hinderlich wäre. Man müsste das mit den entsprechenden Fachleute besprechen und ausprobieren.

RR: Wann soll der Mitarbeiten im Prozess über eine potentielle Beschwerde informiert werden? Soll diese Information dem Mitarbeiter angezeigt werden sobald der Kunde anruft und er identifiziert wurde? Oder ist es wichtiger, dass der Mitarbeite bereits vor einem Anruf diese Information erhält, damit er proaktiv auf den Kunden zugehen kann?

FB: Ideal wäre es wenn man präventiv arbeiten kann. Das heisst, dass man das Problem erkennt bevor der Kunde sich beschwert und man deshalb vorher darauf reagieren kann.

RR: Was gibt es heute für Mechanismen um die Mitarbeiter über Kundenereignisse zu informieren? Was macht der Mitarbeiter mit dieser Information?

FB: Es gibt Aktivitäten, EreigMas (Ereignismanagement) beziehungsweise Leads, welche die Mitarbeiter erhalten. Wenn zum Beispiel ein Kunde Geburtstag hat, so könnte der Mitarbeiter ein Lead erhalten damit er entsprechend darauf

reagieren kann. Oder wenn der Kunde umzieht, so könnte der Mitarbeiter über ein Lead informiert werden, dass er die Police überprüfen muss und wenn nötig mit dem Kunden Kontakt aufnehmen sollte. Solche Prozesse haben wir heute schon im System.

RR: Es wäre deiner Meinung nach sinnvoll die Informationen über eine potentielle Beschwerde auch über einen solchen Prozess zum Beispiel mittels Leads abzuwickeln?

FB: Es wäre sicher ein möglicher Weg das in ein bestehender Prozess zu integrieren.

RR: Jetzt würde ich gerne zu den eher technischen Fragen kommen. Soll das Machine Learning Modell im produktiven System laufend oder nur periodisch dazulernen?

FB: Aus meiner Sicht sollte das System ständig dazulernen. Aber wir haben gewisse Aktivitäten welche periodisch laufen, die eher zu Beschwerden führen könnten, wie zum Beispiel der Mahnlauf oder die Prämienberechnung für das neue Jahr. So gesehen könnte es auch genügen wenn das System nur periodisch dazulernt.

RR: Gibt es deiner Meinung nach Informationen in der Kontakthistorie welche eher einen Einfluss darauf haben, ob das nächste Ereignis eine Beschwerde sein wird? Und gibt es solche Informationen die eher unwichtig sind?

FB: Aus den Informationen die heute in der Kundenkontakthistorie abgelegt werden, ist der Kontaktgrund sicherlich ein wichtiger Faktor der einen Einfluss auf eine potentielle Beschwerde haben kann. Auch der Kontaktkanal kann wichtig sein, denn je nach Kanal ist die Geschwindigkeit der Kommunikation unterschiedlich. Wenn der Kunde einen Brief

schreibt, so erwartet er nicht eine sofortige Antwort, aber wenn er via Kundenloginportal Kontakt aufnimmt, so ist es kritischer, dass eine Antwort möglichst schnell erfolgt. Und natürlich ist es auch wichtig zu wissen ob es sich bei der Kontakthistorie−Eintrag um eine Beschwerde handelt oder nicht. Eine sehr wichtige Informationsquelle kann das Beschreibungsfeld sein, wo der Mitarbeiter völlig frei seine Notizen zum Anliegen machen kann. Wenn man aus diesem Freitext Feld Informationen herleiten könnte, so könnte das auch wertvoll sein. Bei den Informationen welche man auf diese Stufe der Kontakthistorie hat, gibt es nichts was ich nun als unwichtig oder irrelevant betrachten würde.

RR: In der Kontakthistorie wird die ID der verantwortliche Person aufgeführt. Sie beinhaltet die eindeutige Identität des CSS Mitarbeiters der für den jeweiligen Kontakthistorie−Eintrag verantwortlich war. Ist es deiner Meinung nach kritisch solche mitarbeiterspezifische Informationen zu verwenden? Gibt es moralische Bedenken?

FB: Was hier natürlich als Frage aufkommt, ist die Qualitäts−sicherung. In der Kundenservice Center wird sehr viel mit Qualitätssicherung gemacht. Sie machen dort Audits, hören Gespräche ab und auf Grund von dem führen sie gezielte Schulungen mit den Mitarbeitern durch. Wenn man also das Attribut 'VerantwortlicherId' verwendet und das System sollte feststellen, dass bei gewissen Mitarbeitern häufiger Beschwerden auftauchen als bei anderen, so können diese Mitarbeiter gezielt geschult und ausgebildet werden. Man kann es demnach aus dieser Optik anschauen mit dem Ziel die Qualität der Kundeninteraktionen und auch die Mitarbeiter besser zu machen. Man kann es als Unterstützung betrachten.

Listing A.1: Interview with Fredi Bircher

# B. A story with SML

## B.1. The story of customer c0000002

```
1  STORY "Contact history of customer c0000002"
2      DICTIONARY
3          USE "CSS_Dictionary"
4          ENTRY c0000002
5              DEFINITION 1 NOUN "css customer nr. 0000002"
                   HYPERNYM customer(1)
6          ENTRY "15.11.2017 16:39"
7              DEFINITION 1 NOUN HYPERNYM datetime(1)
8          ENTRY "21.09.2017 12:14"
9              DEFINITION 1 NOUN HYPERNYM datetime(1)
10         ENTRY "21.08.2017 10:39"
11             DEFINITION 1 NOUN HYPERNYM datetime(1)
12         ENTRY "10.08.2017 16:39"
13             DEFINITION 1 NOUN HYPERNYM datetime(1)

14     NARRATIVE
15         css(2) contact(1) c0000002(1) by(2) post(2) at(2)
               "10.08.2017 16:39".
16         WHILE
17             {
18                 css(1) send(1) praemienabrechnung(1).
19             }

20         c0000002(1) contact(1) "customer service"(1) on(1) phone
               (1) at(2) "21.08.2017 10:39"(1).
```

```
21          WHILE
22              {
23                  a0000002(1) accept(1) call(1).
24                  c0000002(1) want(1) complaint(1).
25                  a0000003(1) handle(1) complaint(1).
26                  issue(1) of(1) c0000002(1) be(1) solved(1).
27              }

28          c0000002(1) contact(1) "customer service"(1) on(1) phone
                (1) at(2) "21.09.2017 12:14"(1).
29          WHILE
30              {
31                  a0000002(1) accept(1) call(1).
32                  c0000002(1) want(1) "insurance cancellation"(1).
33                  a0000003(1) mutate(1) policy(2) of(1) c0000002
                        (1).
34                  issue(1) of(1) c0000002(1) be(1) solved(1).
35              }

36          c0000002(1) contact(1) "customer service"(1) on(1) email
                (1) at(2) "15.11.2017 16:39"(1).
```

Listing B.1: CSS Dictionary example

# C. CD

A CD with the title *MSc Thesis Rahul Rao* was handed in with this thesis. This CD not only contains the Latex Files and the PDF created for this thesis, but also the source code for the java application *kukohi-transformator* written for preprocessing the data. All files containing the customer data will be omitted from the submission for data protection reasons.

## C.1. Contents of the CD

| | | |
|---|---|---|
| \ | | master thesis document as pdf |
| \electronic-bibliography\ | | electronically available references |
| \experiment-code\ | | Python code for all the experiments |
| \kukohi-transformator\ | | source code for... |
| | src\...\step1\ | ... preprocessing the data |
| | src\...\step2\ | ... preparing the data for ML |
| | src\...\step3\ | ... generating the stories (in progress) |
| \msc-latex\ | | main *.tex file for this thesis |
| | bib\ | bibliographical database file |
| | chapters\ | sections content |
| | images\ | used images |

# D. Statutory declaration

I confirm that I have written this thesis independently. All text passages that do not originate from me are marked as quotations and marked with the exact reference to their origin. The sources used (also applies to illustrations, graphics, etc.) are listed in the bibliography.

place, date: Lucerne, 24.05.2018

signature: .....