Using machine learning for assessing customer suitability in business to business environment

Master's thesis
University of Turku
Department of Future Technologies
Software engineering
2019
Markus Lindberg

University of Turku
Department of Future Technologies

Markus Lindberg: Using machine learning for assessing customer suitability in business to business environment
Master's Thesis, 55 p., 0 app. p
Software engineering
May 2019

The industry has gone through multiple revolutions as a result of advancements in technology. Now, it might be on the verge of another one because of machine learning. In business to business setting, sales processes include many manual tasks. They consume a lot of time, that could otherwise be used for deepening relationships with customers or for acquiring new ones. Machine learning could be used to support and automate these tasks. Also, it could be used to forecast how a sales process ends.

This thesis proposes a new approach for using machine learning in the said environment. Instead of using sales data to just predict the outcomes of active sales processes, it is used to estimate how good a potential customer would be for a company. With these kinds of estimates, a company could more effectively screen the customers making their sales efforts potentially more profitable. The thesis aims to research questions that arise from this approach. These include the question of how to provide suitability estimates without saved data, when to start using machine learning for providing the estimates and how to maintain the performance of selected machine learning method. In the thesis, an open business to business dataset is used. It is introduced and analyzed. Also, a selection of machine learning models is introduced and one of them selected for the thesis. To answer the open questions of the thesis code is developed and tested. The code forms a two-mode system for providing suitability estimates. The purpose of the code was to verify the ideas useful for such a system.

As results of the thesis it can be stated that without saved data it is possible to offer suitability estimates by defining an optimal customer and comparing all the new customers to it. The system can start using machine learning once a classifier fitted to the data gives good enough cross-validation results. Once a machine learning model is in use, its performance can be maintained by retraining the model and finding the optimal parameters each time the dataset has changed. Finally, it is however stressed that the suitability estimates are not absolute truths. They should not be used blindly, but as decision support when selecting new customers for a company.

Keywords: Machine learning, B2B, Decision support

Turun yliopisto
Tulevaisuuden teknologioiden laitos

Markus Lindberg: Using machine learning for assessing customer suitability in business to business environment
Diplomityö, 55 s., 0 liites.
Ohjelmistotuotanto
Toukokuu 2019

Teollisuus on käynyt läpi usean muutoksen teknologian kehityksen seurauksena. Koneoppimisen takia uusi muutos on taas edessä. B2B ympäristössä myyntiprosesseihin kuuluu monia manuaalisia tehtäviä. Niihin kuluu paljon aikaa, jota muuten voisi käyttää asiakassuhteiden syventämiseen tai uusien asiakkaiden hankkimiseen. Koneoppimista voidaan käyttää näiden tehtävien tukemiseen ja automatisoimiseen. Se soveltuu myös myyntiprosessien lopputulosten ennustamiseen.

Tämä työ ehdottaa uutta tapaa käyttää koneoppimista B2B ympäristössä. Sen sijaan, että myyntidataa käytettäisiin uusien myyntiprojektien lopputulosten ennustamiseen, sitä käytetään arvioimaan, kuinka hyvä potentiaalinen asiakas on yritykselle. Tällaisia arvioita hyödyntäen yrityksellä on mahdollisuus tehokkaampaan asiakkaiden seulontaan ja tätä kautta myyntiprojektien kannattavuuden parantamiseen. Työ käsittelee tästä lähtökohdasta nousevia kysymyksiä. Näihin kuuluu kysymys siitä, miten sopivuusarvioita on mahdollista tuottaa ilman talletettua tietoa, milloin koneoppimista voidaan käyttää soveltuvuusarvioihin ja miten ylläpitää valitun koneoppimismallin suorituskykyä. Tässä työssä käytetään avointa B2B-aineistoa. Se esitellään ja analysoidaan. Kokoelma koneoppimismalleja esitellään myös, jonka jälkeen yksi niistä valitaan. Tutkimuskysymyksiin vastaamiseksi ohjelmakoodia rakennetaan sekä testataan työssä. Koodi muodostaa kaksiosaisen järjestelmän soveltuvuusarvioiden tuottamista varten.

Työn tuloksina voidaan mainita, että on mahdollista tuottaa soveltuvuusarvioita vähällä määrällä talletettua dataa määrittelemällä optimiasiakas ja vertaamalla muita asiakkaita tähän. Arvioiden tuottamiseen voidaan ruveta käyttämään koneoppimista, kunhan tämä malli dataan sovitettuna tuottaa riittävän hyviä tuloksia ristiinvalidoinnilla. Kun koneoppiminen on käytössä, sen suorituskykyä voidaan ylläpitää uudelleenkouluttamalla käytössä oleva malli ja hakemalla sille optimaaliset parametrit aina, kun käytetty data jotenkin muuttuu. Lopuksi työssä kuitenkin painotetaan, että tuotettuja arvioita ei tule pitää ehdottomina totuuksina. Niitä ei tulisi käyttää sokeasti, vaan päätöksen tukena uutta asiakasta valittaessa.

Avainsanat: Koneoppiminen, B2B, Päätöstuki

Contents

1 Introduction

During the industrial age, the industry has gone through multiple revolutions. First revolution was started by mechanization and the use of steam power. It brought changes with it making tasks previously dependent on human and animal work much more effective. The second revolution introduced mass production with the availability of electricity. The third introduced automation and computers. Now that the computational power of the computers has increased and there is great amount of data available in the internet, the industry might be on the verge of another revolution. It is caused by machine learning.

Before discussing the effects of machine learning on industry any further, two concepts need to be introduced. Selling is naturally involved, when it comes to industry. A company can manufacture products or services and sell them directly to their customers or other businesses. The way in which the demand forms for these manufactured goods can be seen as a division between two types of markets. If a business manufactures products or services, that are shaped by the specific needs of the buyer and sells these products straight to the consumer, this is called business to customer marketing (B2C). If the demand for a product is somehow derived from the needs of a further customer, this is called business to business marketing (B2B). First this activity was known as industrial marketing. It involved the sales and transactions of raw materials that other businesses used in their daily activities. With the growth of service and technology sector, the name was replaced with B2B marketing. This name describes all relationships that generate value between different businesses, government agencies, non-profit organizations and people representing the said agencies and organizations [1]. This thesis will not focus on activities that occur in business to customer setting. However, technologies used in this setting may be mentioned.

Sharma and Syama [2] introduced, how the newest revolution would affect the sales process of each company. The process in general includes a lot of manual tasks that the sales representatives need to do. As a result of this, they do not have as much time to focus on building customer relationships. Machine learning could be used to automate and support these manual tasks. In sales process, generating and validating leads is one of the time-consuming activities. The study mentioned one estimate, according to which the sales representatives are using 80% of their time validating leads. According to the article it has also been demonstrated that machine learning can be used in lead generation. Using it also reduces the workload required from sales personnel of a company.

Decision making could also benefit from machine learning. Bohanec et al [3] mentions that companies relying on data-driven decisions are more productive and profitable than their

counterparts. It also mentions studies related to decision support systems. Those who use this kind of system, are more likely to use the recommendation produced by it, if explanations for the produced recommendation are available. This introduces one problem: machine learning models, which give the best results, are not transparent. This leads to low usage and acceptance by the users. In both papers of Bohanec et al [4] and Bohanec et al [3] a machine learning model was used to forecast the outcome of a sales process in business to business environment. Although Bohanec et al [4] mentioned, that there have been great advancements in the practice of forecasting sales, the decision makers are reluctant to start using any recommender systems. Instead, they trust their own mental models, which can lead to sub-optimal forecasts.

In Bohanec et al [4] the machine learning model was used in a process, where two sets of predictions were made on the outcomes of active sales processes. First set was produced in a meeting based on human evaluation and the other was made with the help of a machine learning model. These machine learning predictions were produced by an external consultant. One of the key problems of sales personnel mentioned previously, is the extensive time needed to qualify leads. Therefore, one of the interests of this thesis would be to see, if the process mentioned in Bohanec et al [4] could be further automatized.

In the same study, it was also mentioned that business environment is changing fast. It proposes a challenge for predicting sales with machine learning. New factors might be introduced to the sales process so that the old model is unable to provide accurate predictions. This is called the concept drift. According to the study, it is countered by continuously reflecting on the predictions produced by used machine learning model and taking action accordingly for example by adding a new variable to the data. In this thesis, ways of dealing with concept drift will be studied.

So, in short, machine learning is used mostly as a forecasting tool for sales projects. Knowing how and when a sales project ends, can help to steer a company's operations internally. However, instead of just focusing on the sales projects and their outcomes, the information on previous sales projects could be used to evaluate a customer even before any sales activity takes place. It would refine the quality of sales projects, save time and function as a decision support system for companies assessing new customers. This thesis will survey how the process of creating a machine learning system for assessing the suitability of a new customer can be automatized. Suitability of a customer is a term used in this thesis to describe how likely it is that a sales process started with the potential customer is going to end favorably for the company. Also, the ways in which the possible concept drift can be dealt with, will be studied. Finally, a machine learning system will be designed and developed as a proof of concept. It is to show the capabilities of the selected machine learning

method and the ways in which the automatization was possible in the end. The aim is to build the system so that its ideas could be later taken into use by a company.

The research questions of the thesis can be summarized as follows

1. How can machine learning be used to provide suitability estimates for potential customers?
2. How much can be automatized from the creation and use of machine learning models in customer suitability estimations?
3. How can the effects of concept drift be automatically dealt with?

This chapter of the thesis introduces the overall topic. In the second chapter, a literature review will be conducted to gain insight in the ways machine learning has been used in business to business environment. The third chapter will introduce the used data in the thesis. The fourth chapter will introduce machine learning in general and include general descriptions of some popular machine learning models. In the fifth chapter, cross-validation will be introduced. Also, nested cross validation will be performed on the data to find the most suitable model for the system. Descriptions of technical details of the conducted cross-validation will be included in the chapter. In the sixth chapter, the system to be built will be described on a general level. Seventh chapter will describe the technical details of the proposed system and demonstrate each part. Chapter eight contains conclusions and further work.

2 Literature review

In this section, a literature review over technologies used in B2B marketing will be conducted. After the review, a summarizing chapter will follow. Summary will be used to gain an overall idea of what have been done in the field and what could be used in the system to be developed later in this thesis.

A lot of research exists regarding predicting sales and their outcomes. In the paper of Bohanec et al [3] the researchers focused on sales prediction in B2B setting. For the prediction they developed a set of attributes that described a sales process. With them they used a random forest to predict the outcome of the project. They also tested other machine learning models like decision trees and support vector machines for the same task, but validations with classification accuracy and area under curve justified their selection of random forests. In their work, the researchers also made effort to explain, which of the attributes used are the most influential. They used methods called EXPLAIN and IME, since random forest is a black box model and not interpretable like decision trees. The mentioned methods are explanation methods that have sensitivity analysis at their core. In such analysis, an attributes effect on the output is investigated by simulating the lack of it. If the change in the output is big, the attribute is important.

In Bohanec et al [4] the user acceptance of machine learning models was addressed. Multiple machine learning models were evaluated, but in the end random forest was selected for their study. They also used IME and EXPLAIN model explanation methods in their study. Action Design Research (ADR) model was implemented in the study to promote the user acceptance of the developed machine learning predictor. In ADR, users are involved in the development of the predictor. They co-operate with researchers in an organizational setting to develop the solution to be used.

Chen and Lu [5] studied ways in which sales of computer resellers could be predicted. They used three types of products, namely desktop computers, LCD -screens and notebook computers as example products in their study. The biweekly sales data from January 2005 to September 2009 was used. For prediction, they proposed multiple different combined clustering and machine learning techniques the best of which was growing hierarchical self-organizing map with extreme learning machine. The overall idea in their predictor was to cluster their training data and then use the data from these clusters as training data for extreme learning machine.

The paper of Gentner et al [6] states that B2B companies are behind B2C companies, when it comes to analyzing the customer base for needs, affinity for technology and future customer applications.

According to them, this analysis is still based on management heuristics and does not cover the complexity of customer relationships. In the study they developed forensic customer approach for the identification of weak signals. Weak signals are rare events or indicators of change. In the application, they developed this said framework within an international German company, which developed equipment for hydro powerplants. The product management team was preparing to launch a new sensor. They used CRISP-DM, a cross industry standard process for data mining in the project. For the training of classifiers, they used data from the company ERP and CRM. The data consisted of 23 different variables and for the training of classifiers data instances that had more than 15 variables missing were rejected. The study used Resilient multilayer neural network and a c4.5 decision tree algorithm. The decision tree achieved slightly better classification accuracy than neural network.

The study of D'Haen and Van den Poel [7] proposed an online tool to help acquire customers. According to the study acquiring customers in B2B setting can be difficult because of the amount of available data. Sales force of a company can lose a lot of time, while pursuing bad business leads. Because of this a tool was developed. Its purpose was to automate the work of the sales force and to generate a list of prospects that are more likely to turn into leads and ultimately into customers. Prospects mean potential customers that meet certain criteria. Leads mean prospects that will be contacted and are most likely to respond. The proposed tool works in three phases and uses the current customer base of a company as a basis for generating the prospect list. In the phase one, k-nearest neighbor algorithm was applied. It was used with a Jaccard similarity coefficient to cluster prospects with profiles of current customers of the company. In the center of a cluster, there is a profile of an existing customer. The closer the prospect is to the center of a cluster, the more likely it is going to turn eventually into a customer. The result of the first phase is a list of prospects each of which having a similarity value ranging from 0 to 1.

The phase two uses the list generated by the first phase. For each item in the list, a predicted probability was added. For this, they included both logistic regression and decision trees in their system. AUC was calculated for both models and the one with higher value was used in the end. Artificial neural network was used as well, but only as a backup due to it being slow. Phase three in turn is a combination of phases one and two. It calculates weights for both phases and produces the final prospect list. This list is used as feedback in phase two.

Bohanec et al [8] aimed to promote the acceptance of machine learning models in business. The study claimed that companies basing their operations on data-driven decisions were more profitable and productive than their counterparts. They also stated that those using knowledge-based systems

were more likely to act on the recommendations of the system if explanations were also available. In their study they developed a framework that integrated the historical data owned by the company into their decision-making process. People were included in this process. They developed this framework using ADR for a real medium-sized company selling software solutions. For the framework, they tested multiple different machine learning models. They were evaluated with classification accuracy (CA) and area under curve (AUC). Random forest produced the best results, but for demonstration purposes of the framework, they used also other models. EXPLAIN and IME was used to explain the outputs of used machine learning models. The company CRM was also updated as a part of the study with new variables. This resulted in a machine learning dataset used in the study. The set was also made publicly available.

The process of the framework starts with a meeting. In the meeting the participants produce an initial forecast of sales leads that are supposed to be closed as won after a month. This forecast is updated to the company CRM. Then an external consultant uses a selected machine learning model on the updated CRM and produces another forecast. It is then shown to the company personnel with explanations. Based on this, the company then updates and re-adjusts their initial forecasts. At the end of the period of the framework the CRM is again updated.

D'Haen et al [9] focused on customer acquisition process of a company which they state, is a stressful undertaking. One part of this acquisition is trying to predict, which of the leads are profitable, once they turn into customers. For this, most companies use external data sources. The study however points out that these sources of data can produce low quality datasets. Another way to acquire data for such predictions, is to mine information from company websites. According to the paper, this is rarely done in companies. Both ways of acquiring information were evaluated with different data-mining techniques, namely logistic regression, decision trees and bagged decision trees. The aim was to find the best working combinations. In their study, they selected data from the point of view of a German B2B mail order company. Companies, that had a website and were in Germany, were selected from commercial data sources. The ones, that were profitable for the mail order company and had a website, were used as the sources for web data mining. As a result, they found out that bagging trees worked the best out of the tested mining techniques. They also found out that data mined from the web results in better predictions than commercial data. However, combining the two sources resulted in even better predictions. When using the two data sources together they noticed that bagged trees outperform the regression analysis and normal decision trees. When the bagged trees were evaluated against both available data sources, predictions with

web data were much better than with commercial data. Also here combining the two sources resulted in even better predictions.

Bohanec et al [10] created a list of machine learning attributes for B2B setting. The list was meant to support knowledge engineers in data structure construction of sales opportunities for machine learning. As machine learning models are highly dependent on the input data quality and chosen parameters, it was the aim of the study to create a list of comprehensive parameters for said setting. They reviewed previous study and academic papers. Over 75 attributes were selected for the list and categorized to five different categories. These categories were as follows:

1. Economy of the client: Describes the situation of client
2. Individual seller: Describes the traits of a seller
3. Internal: Describes, how an opportunity is perceived in the company
4. Relationship: Describes the nuances of B2B relationship
5. External: Describes the external economic situation

At the end of the study, it was stated that from all the listed attributes, a subset that provides a solid description of sales opportunities can be selected for each company based on expert opinion.

Meire et al [11] integrated data extracted from social media pages into B2B customer acquisition decision support system. According to the study, the potential of social media has been recognized in business to business environment. However, its adoption rate has been much slower than in business to customer environment. In total, three different sources of data were used in the study, namely commercial data, data from prospects websites and data extracted from company's Facebook page. The aim of the study was to evaluate the predictive value of the data from company Facebook page. It was compared with the data extracted from company website and with data from a commercial source. For the evaluation, the sales funnel model of D'Haen et al [9] was used. Random forest classifier was selected for the classification tasks of the study. Different combinations of the mentioned data sources were used as input data in both phases of the acquisition model. The predictive performance of the data was evaluated with AUC and Lift. The study found out that the model performance was only slightly better if commercial data was used. So, they argued, that it might be worth building the customer acquisition system with only data acquired from the websites and social media.

Jadhav and Deshpande [12] reviewed multiple techniques to detect concept drift. Concept drift means a change in the distribution of data. The study states that when concept drift occurs, the quality of predictions offered by classifier trained on data with initial distribution starts to worsen.

In their study, they form a distinction between two types of concept drifts, namely gradual and sudden. Sudden concept drift occurs when data with a distribution "n" gets replaced with a distribution "m". Gradual concept drift means that instances from datasets with two or more distributions appear at the same time and over time, the probability of encountering instances from an unknown distribution increase. In the paper, online models and block-based models are introduced and their capabilities in detecting concept drift mentioned. The study states that online models are better at detecting sudden concept drifts as they are updated each time a new data instance is made available. Block based models on the other hand can detect gradual concept drifts better as their updates are done with a set of new data instances. They also highlight that online models are not good in detecting gradual drifts and block-based models are not good in detecting sudden drifts. Because of this, an ensemble system for detecting both types of concept drifts equally well is proposed. The system would keep both online and block type classifiers in an ensemble with a buffer for data instances. These classifiers would be weighted. Once a data instance is entered in the ensemble, the online classifiers of the ensemble are used to predict the class label for the instance. All the predictions are aggregated to get a majority vote. In addition to this operation, the online classifiers are updated, and their weights are modified based on accuracy of previous predictions. After this, the data instance is stored in the buffer. Once it is full, similar operations are carried out with the block classifiers. The weights of the block classifiers are modified based on the accuracy of newest block of data, the block classifier is updated based on the newest data block and the class labels for instances are produced as a majority vote. Error rate is calculated for both classifier types, and once it falls below a certain threshold and stays there for the next blocks of data, concept drift is detected.

Bruno et al [13] proposed an ensemble system called DDE for detecting concept drift. Their motivation was to achieve more precise detections of concept drifts by combining three concept drift detection methods. Different methods for detecting concept drift were reviewed, the following of which ended up in their proposed system: DDM, ADWIN, ECDD, HDMMa and HDMMw. DDM detects changes in a distribution. It does this by analyzing the probability of error (p) and its standard deviation (s). Both values are calculated for each instance inserted in the system. The method also maintains knowledge of minimum values for error (p_{min}) and standard deviation (s_{min}). Drift is detected, when $p + s \geq p_{min} + s_{min} * \alpha$, where α is the confidence level.

ADWIN takes advantage of a sliding window technique. The window size increases while the data distribution remains the same and decreases while concept drift occurs. In ADWIN there are also dynamically adjusted sub-windows for older data and recent data. The difference between the

averages of data from both windows is calculated and concept drift is detected, when the difference is higher than a set threshold.

HDDM is a collection of methods that measure values available during the learning process of a classifier. Drifts are detected, when a significant change is detected from these values. "**Test_a**" variation of the method uses moving averages while "**test_w**" variation uses weighted moving averages. ECDD uses estimations of mean and standard deviation of variables and detects changes from that. All these methods were applied in DDE. Three different sensibility levels were proposed and different combinations of introduced concept drift detection methods were attached to each of the levels. Each level sensibility defines how easily a concept drift is detected. If one of the methods notices a drift, it is immediately confirmed by the method at sensibility level one. If sensibility level is higher than one, only warning is issued at the drift detection of one method. When two or more methods detect this drift as well, the ensemble signals drift. Max wait is also defined by DDE. The detection of concept drift is classified as false positive, if one detector detects a concept drift that is not confirmed by other detectors within the wait period. The aim of the method was also to be lightweight. That is why they had only one classifier in use when the detected concept is stable and two for other states.

In Costa et al [14] the methods for detecting concept drifts were divided into two groups, namely blind and informed. Blind methods do not try to explicitly detect a concept drift. Instead, they counter them by regularly retraining the used machine learning model. Informed methods on the other hand have a mechanism for detecting the drift. In the study they proposed a method for detecting concept drift explicitly and in unsupervised manner. By using this way of concept drift detection, they aimed to provide an alternative to other state of the art concept drift detectors for example DDM and EDDM. Those concept drift detection methods are incremental and require that the true labels of the data instances are readily available. According to the study, this is an unrealistic requirement when it comes to many real-world applications. In the paper DDAL method was proposed for concept drift detection. It works in two phases and uses blocks of data. In the phase one a classifier is generated from data, which is currently available. Then in the second phase the density variation of the most significant instances is monitored. The most significant instances are identified by using a class-separating hyperplane and user defined virtual margins. If the density variation of these most important instances is higher than user defined threshold, drift is detected. If this is the case, reaction module of the method is triggered. There a new classifier is trained with the newest block of data. This classifier replaces the old one. If no drift is detected, the newest batch is classified, and the process returns to detection phase.

2.1 Summary of the review

In the articles found for this thesis, the focus was to develop a system to help either with finding the right customers for a company or with predicting the outcome of a sales project. For this, machine learning was applied. Many of the studies evaluated multiple models but for example in Bohanec et al [3], Bohanec et al [8] and Meire et al [11] random forest was used.

Explanation methods of machine learning models came up in the articles. Bohanec et al [8] stated, that companies who use knowledge-based systems, are more likely act on the recommendations produced by the system if there are explanations available. In Bohanec et al [3] explanation methods EXPLAIN, and IME were introduced. They were applied to produce explanations for the predictions of machine learning model in use. In D'Haen et al [9] data mined from the internet was considered as an alternative to commercially purchased data. Both sources were evaluated. In the end, it was noticed that combining data from both sources improved the predictions of used machine learning models. In Meire et al [11] used data extracted from social media in a customer acquisition decision support system. D'Haen and Van den Poel [7] developed a three phased online tool for customer acquisition. In the first phase the potential customers were analyzed based on existing customers, in the second phase, probabilities were attached to the customers and in the last one, the final analysis was done for the potential customers.

Detecting concept drift is an important factor to consider. In Jadhav and Deshpande [12] and Bruno et al [13] employed an ensemble-based approach. Jadhav and Deshpande [12] introduced both sudden and gradual concept drift. In the study it was stated that the gradual drift would be better detected with a block -based machine learning models. Sudden drift on the other hand would be better detected by online learners. So, for the detector, both types of learners were combined. The actual detection of concept drift would be done by monitoring the error rate of the classifiers in the detector. In Bruno et al [13] an ensemble method was also proposed. The main idea of this method was to combine three different drift detection methods in a three different sensibility levels. The higher the level the longer the detection of drift would take. At level one, drift would be signaled already with one method detecting a drift. At higher levels, drift detected by one method would cause only a warning. When other methods of the level detect the drift as well, the drift is verified and signaled by the ensemble. In Costa et al [14] drift was detected by monitoring the changes in density variations of the most significant data instances. If the variation was greater than a user defined level, drift was signaled.

When it comes to this thesis, the system constructed in D'Haen and Van den Poel [7] would seem the most promising. The focus there is on automation, and unlike in Bohanec et al [8], no external consultant is used to produce predictions with a machine learning model. Because of this, there is potential for saving time. As reported in Bohanec et al [3], Bohanec et al [8] and D'Haen et al [9] multiple machine learning models were somehow evaluated and tested after which the best one was selected. So, for this thesis the same procedure will be used. Later, the results of the procedure will be discussed more in depth. Here however it can be stated that further discussion of explanation methods will be excluded from this thesis. Bohanec et al [10] created a list of attributes that would best represent a sales process between two companies. Because using quality data with machine learning is important it is recommended to take those attributes in to account when inserting data for the system of this thesis.

3 Introducing the dataset

In this chapter, the business to business dataset from Bohanec et al [8] will be introduced. It will be analyzed on a general level. Principal component analysis will also be introduced and then performed on the dataset. This is done to see how different classes of the dataset are positioned in relation to each other and if there are any visible boundaries between them. Later in the thesis the instances within the dataset will be used as a test data when constructing the system for validating new customers.

3.1 Dataset

The selected dataset contains real world B2B business cases of a medium sized company selling software solutions. These business cases will be referred to as "instances". The dataset will be referred to as "B2B dataset" [15]. The set has 448 instances, each of which has 22 + 1 attributes. 49% of the instances are labeled as lost and 51% are labeled as won. The attributes that all the instances have were selected by the company at a workshop during the study. Next, all the selected attributes will be listed, and their meanings explained:

- Product
    - Offered product
        - Product A - K
- Seller
    - Seller's name
        - Seller 1 - 20
- Authority
    - Authority level at client side
        - Low, Mid, High
- Company size
    - Size of a company
        - Small, Mid, big
- Competitors
    - Does the company have competitors?
        - Yes, No, Unknown
- Purchasing department
    - Is the purchasing department involved?
        - Yes, No, Unknown

- Partnership
    - Is the sale occurring in partnership?
        - Yes, No
- Budget allocated
    - Did the client reserve budget?
        - Yes, No, Unknown
- Formal tender
    - Was formal tendering procedure required?
        - Yes, No
- RFI
    - Was a request for information received?
        - Yes, No
- RFP
    - Was a request for proposal received?
        - Yes, No
- Growth
    - The growth of the client company
        - Stable, Growth, Unknown
- Positive statements
    - Were positive statements expressed?
        - Yes, Neutral
- Source
    - Source of the project
        - Referral, Joint past, Unknown
- Client
    - Type of a client
        - Current, New
- Scope clarity
    - Has the scope of the implementation been defined?
        - Clear, Few questions
- Strategic deal
    - Does the deal have strategic value?
        - Unimportant, Average important, Very important

- Cross sale
    - Is the company selling additional product to an existing customer?
        - Yes, No
- Up sale
    - Is the company selling more expensive product or upgrade to the customer?
        - Yes, No
- Deal type
    - What is the type of the deal?
        - Project, Maintenance, Solution
- Needs defined
    - Did the customer clearly state their needs?
        - Yes, No, Info gathering
- Attention to client
    - Attention to a client
        - Strategic account, Normal, First deal, Bad client
- Status
    - The outcome of the sales project
        - Won, Lost

The attributes presented here come from a larger list of attributes found to characterize business to business sales. The list was constructed by Bohanec et al [10]. In their study, the attributes were split into five categories. Those categories were presented in the literature review of this thesis. In this set, attributes from following categories are present: Internal (product, source, cross-sale, up-sale, strategic deal, scope clarity and client status), Individual (seller), Client-related (company-size, competitors, RFI, RFP, budget-allocated, formal tender, authority, growth, positive statements), Relationship (needs defined, attention to client). Partnership, purchasing department and deal type were parameters added by the company for which the set of parameters was developed in Bohanec et al [8].

Oshiro et al [16] introduced density-based metrics for analyzing the difficulty in which a dataset can be learned by a classifier. They propose that the density of a dataset can be calculated with a similar formula used for physical objects. For them, density is calculated by dividing the mass of the object by its volume. For a dataset, the study considers the number of instances as the mass of a dataset. The volume is given by the dataset's attributes.

In the study, they proposed three ways in which the density could be calculated. Only one of them was used. The formula considers three variables, namely classes of a dataset (c), attributes of a dataset (a) and the instances of a dataset (n). The used formula was derived from its original, physical object inspired form as follows:

$$D_1 \triangleq \frac{n}{a}$$

$$D_2 \triangleq log_a n$$

$$D_3 \triangleq log_a \frac{n}{c}$$

The first formula is the starting point of the dataset density calculation. In the study, the density was calculated for 29 different datasets. In all those datasets, the considered values vary understandably. Because of this, the values were considered in a natural logarithmic scale. This led to the second formula. When the effect of the number of classes on the instances was considered, the result was the final formula. This formula was the one used in the density calculations of the study. In it they determined that if the result of the calculation was lower than one the dataset had a low density and it would probably be difficult to learn from the dataset. The density was considered high if it was one or more. In this case, the study suggested that it might be easy for a learner to learn from the dataset.

As mentioned before, the dataset has 448 instances, 22 attributes and two classes. As an evaluation measure, the density of the B2B dataset was calculated using the formula as follows:

$$log_{22} \frac{448}{2} \approx 1.8$$

The resulting value would indicate that the dataset has a high density and learning from the dataset might be easy.

3.2 Principal component analysis

As stated before, the B2B dataset has 22 attributes in total per instance. In theory these variables could be plotted against each other for a series of graphs. It however is effective for datasets that have only few variables. If there are more, like in the case of the B2B dataset, new ways of analysis are needed to effectively gain insight in the dataset.

Reris and Brooks [17] mentioned a paper of Karl Pearson named “On Lines and Planes of Closest Fit to Systems of Points in Space”. The ideas in the paper created the foundation for technique that

would later be called principal component analysis. It is a technique that can be used to reduce the dimensionality of a multi-dimensional dataset. Reduction is achieved by creating a set of new variables that are linear combinations of the original variables.

For example, let us assume, that there is a n dimensional dataset. This dataset can be presented in a n-dimensional space. Once visualized, the set might appear as a large cloud of points. If the cloud is elongated in some direction, it is an indication that two or more parameters are correlated. Principal component analysis tries to find these directions. It starts from the dimension *p*, then moves onto dimension $p - 1$ until all dimensions are analyzed [18]. By doing this it reduces the dimensionality of the original dataset without losing much of the information contained in the data [19]. The found directions are used as axes for the parametrization of the p-dimensional space. These are called the principal components [18]. In other words, the directions are linear combinations of features contained in the original dataset projected in the direction of greatest variance. They are also uncorrelated. The first principal component is situated in the data so that it has the greatest possible variance of the analyzed dataset. The second principal component is calculated so that it needs to be perpendicular to the first component. Its variance is also lower than the variance of the previous one [19].

The analysis of the B2B dataset was continued by performing a principal component analysis on its data. Since the dataset contains only categorical values, the dataset was encoded. The encoding process was carried out for each column or in other words, one attribute at a time. First the categorical values in a column were transformed into numerical values. For that, the Label Encoder from Scikit-learn preprocessing package was used. The encoder gives an integer number for the categorical value that is between 0 and n-1, where n is the number of different categorical values the attribute can have. After that, label binarizer from the preprocessing package was used to one hot encode the column. The encoder carries out its operation in one vs all-style. Finally, PCA from Scikit-learn decomposition library was used to carry out the analysis. The algorithm was set to process the dataset so that only two features were left in the dataset for each instance. This was done to enable displaying the data in a two-dimensional space. With the analysis done, the dataset was visualized as a scatterplot using Pandas data-analysis library.

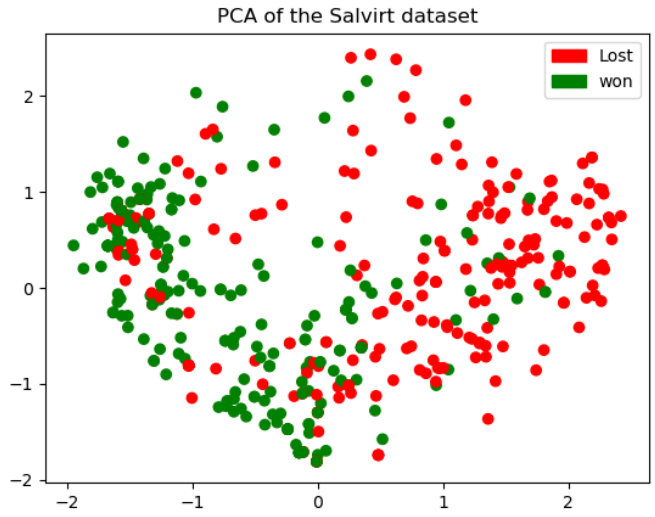

Figure 1 Principal component analysis of the B2B dataset

Each dot on the scatterplot represents an instance of the original dataset. Immediately it can be seen, that the points representing the "Won" and "Lost" – classes overlap in the scatterplot. However, the points representing the classes are not evenly placed. On the left side of the plot, dots of the "Won"-class are more common and more densely placed than on the right side. On the right side on the other hand, the points of "Lost"-class are more common. In the middle of the plot, there is a noticeable area where the density of the points is quite low. Although the classes are not cleanly separated, a vertical line could be imagined somewhere between 0 and 1 on the right side of the plot. This line would divide the cloud of points into regions corresponding to the classes of the dataset. Overall, the plot looks promising and would indicate that a machine learning model could learn from the dataset the plot represents.

4 Machine learning

In the context of sales, a variety of different machine learning methods have been studied. These methods are also application area specific so available options need to be reviewed and the best one selected. The system, that will be constructed as a part of this thesis, will study ways in which machine learning can be used to provide recommendations on how well a company is suitable as a customer for another company. The machine learning model will be trained on data gathered from previous sales processes of a company. A sales project starts with one potential customer and attributes that the potential customer has. The project then will end after some time as won or lost with gained or lost profit. Like stated earlier in the thesis the sales projects can be different for each company. Therefore, the selected machine learning model needs to be able to adapt to the conventions of each company.

In this chapter, different training paradigms for machine learning will be introduced. After that a selection of different machine learning models will be described. In the following chapter the performance of each model will be evaluated with the B2B dataset. One of the models will be selected for the application of this system.

4.1 Learning paradigms

Machine learning offers a different angle for controlling the operation of a computer. Traditionally a computer requires a specific set of instructions that are translated into a machine-readable format for the computer. Based on this translation, the computer can do what it has been told to. These instructions need to be comprehensive and precise. Otherwise the execution of a program might end up in a crash or in an unexpected state. Machine learning gives a computer the ability to learn without being explicitly programmed [19]. Instead a machine learning model needs to be trained for it to be able to function properly. Once the training has been done, the model can find and characterize hidden relationships within a large collection of data. Predictions are also possible.

There are multiple methodologies for training a machine learning model. They all try to optimize some sort of metric that tells about the performance of the overall model [20]. Most of the machine learning applications utilize supervised learning. The training method needs a set of input values and a set of expected output values as training data. During the training the model aims to learn the

mapping between the two. If such a model has been trained well enough, it will be able to provide mappings even for instances of data that have never been inserted in the trained model before [20, 19]. If, however the model is too complex, overfitting may occur. In that situation, rather than generating a general predictive rule for the mappings in similar data, the model knows only what is in the training data [21].

Unsupervised learning is another way of training a machine learning model. Such training needs a set of input data. However, labels or target values are not used in the training. Unsupervised learning can be used for several purposes, such as clustering and extraction of features. Generally, it tries to find hidden structures within the raw data. It does all this without external instructions. [19, 22].

Other training methods include semi-supervised learning and reinforcement learning. Reinforced learning works by trying to maximize a cumulative reward. Essentially it is a trial and error paradigm that gets either rewarded or punished based on certain sequence of actions. In reinforced learning there are some elements to be mentioned. First, there is a policy that maps actions to the perceived state of the environment. Then, there is critique, which is an estimated value function. It criticizes made actions according to the policy of the model. The critique continuously shapes and corrects the policy of the model. Finally, the reward function estimates the perceived state of the environment for an attempted control action [19].

Semi-supervised learning operates somewhere in between supervised and unsupervised learning. Since the data available is commonly unlabeled using supervised learning methods can be difficult. Most of the semi- supervised approaches aim to design an algorithm that takes both labeled and unlabeled data into account. In addition to this, a supervised learning algorithm can be improved by utilizing unlabeled data. This is called the semi supervised improvement. Semi- supervised algorithm can satisfy two types assumptions. If the algorithm assumes that data samples with high similarity must share the same label, the algorithm satisfies the cluster assumption. If the algorithm satisfies the manifold assumption, the algorithm utilizes the fact that the input data lies in the low dimensional manifold of the input space [23].

4.2 Decision trees

Decision trees are composed of decision nodes, branches and terminal nodes. Each decision node contains a logical test that can split the data into two or more classes. Branches connect the nodes to each other so that parent is connected to a child node. Connection made with a branch represents a logical "AND" operation. Each terminal or leaf node represent a class within a decision tree [24]. A data item is classified with the decision tree by going through a path from the top of the tree towards one of the terminal nodes. The path is dictated by the properties of the data item [25].

Decision trees are grown or induced by adding decision nodes to it incrementally [25]. There are multiple algorithms that can be used to build a decision tree. These include for example "ID3", "ChAID" and "See5". Among the most used algorithms is CART. The algorithm operates in two phases, where in the first one the data is split with successive partitions. The generated tree is then pruned to simplify the generated model. This improves its generalization capability [24].

There are several advantages to decision trees. Getting the knowledge from pre-classified examples lowers the need for knowledge from domain expert. The trees can also be used on deterministic or incomplete problems [24]. They are sometimes also much easier to interpret than for example neural networks. They also naturally support such classification problems that can have multiple classes. On the other hand, If the input data changes even a little bit, it can result in major changes to the generated tree [25].

4.3 Random forest

Random forest is a black-box machine learning model Bohanec et al [3]. It is also an ensemble learning method that uses a large collection of decision trees. The trees are grown by using a randomized tree building algorithm. Together the trees form a strong classifier [25, 19]. The resulting classification of the random forest is the class voted by the majority of the trees in the forest [26]. This results in better predictive performance than for example that of a decision tree. Decision trees suffers from high bias and variance [19]. Other advantages of random forest are for example its capability to handle noisy data well, the ability to give an estimate on how relevant a variable is for classification and its robustness to overfitting [27, 19].

In Awad and Khanna [19], growing and using the random forest is described as follows: to grow a specified number of trees for the forest, n number of so-called bootstrap samples are selected from

the dataset. For each of these samples, a classification tree is grown. M prediction variables from all the variables of the dataset are selected at random. Then at a node of a decision tree, a variable that best splits the dataset in two is selected. This variable performs splitting the data in the said node. The next node of the tree then selects another collection of m-variables and again selects a variable that best splits the dataset. Once all the trees have been grown, the resulting forest can be used for predictions.

A relevant question with random forest is the suitable amount of decision trees in the forest. It was studied by Oshiro et al [16]. The study states that most appliers of random forest find the number of decision trees through trial and error. Also, it was mentioned that in some cases by increasing the number of trees in the forest only the computational power needed to run the algorithm increases while little to no performance is gained. In the study 29 different datasets were used. These datasets were grouped into low-, and high-density ones and then used to test random forest with exponentially increasing number of trees. For evaluation, ROC curve and percentage of used attributes in random forest was used. The significance of changes in results with different number of trees was determined with Friedman test. Based on the tests it was noticed that there was no significant difference between forests, when increasing the used trees from 128. The study also noticed that median and mean AUC-values did not change much starting from 64 trees. Finally, the study suggested that to get the best possible balance between AUC, processing time and memory usage, a range of trees between 64 to 128 should be used in a random forest.

4.4 Neural networks

Neural networks are inspired by the human brain and the way in which it processes information. In a network there are neurons that are connected to other neurons via synapses. Synapses have weights to them that indicate the strength of the connection. Modifying these weights causes the network to learn. A synapse takes data to a neuron after it has been multiplied by its weight. Neuron then sums together all its inputs. After this summation the neuron produces an output that usually is the logistic sigmoid of the generated sum [28].

There are many types of neural networks. The networks are considered black-box machine learning models [3]. The most common ones are the feed forward neural networks and recurrent networks. In the feed forward version, the data moves from the input layer of the network to its output layer. There the output of the network is generated. In addition to the input and output layers there can be one or more hidden layers. In the recurrent network the data can move in both directions [29].

The feedforward networks can be trained using the so called “Back propagation” algorithm. It is a supervised learning method that teaches the relationship between the input space and output space to the network [29, 30]. At a general level the training starts so that the synapses of the network have random weights between 1 and -1 [19]. Data is fed to the network with the initial, random weights and then the values obtained from the output layer are compared with the expected results. After this the weights in the synapses are updated and data is then fed to the network again. This is continued until the network produces expected results [29].

One of the problems, that neural networks have, is the problem of local minimum. The gradient descent algorithm will stop, when a minimum in the error is reached. However, there is no guarantee that the reached minimum really is the global minimum of the error. To counter this problem a momentum term can be added to the equation, which updates the synaptic weights. This gives the error some ability to escape the local minimum. Other way of avoiding the local minimum would be to train the network multiple times with different initial random weights in the synapses. Then the solution to which most of the training iterations converge, could be taken as the best solution [30].

When applying neural network, important is to decide on the topology of the network. According to Bailer-Jones et al [30] it is a matter of experience. However, there are some ideas that can be utilized when designing the network. In the study, a theorem is mentioned according to which any continuous function can be approximated with such a network that has only one hidden layer and enough neurons in the network. This also is mentioned in Hill et al [28] and in Matijevic et al [31]. Bailer-Jones et al [30] also mentions in addition to the presented theorem that there are many problems for which a network with one to two hidden layers and 5 to 10 neurons in each is enough. These are however not the only ways of determining the topology. In Matijevic et al [31] multiple different methods for determining the topology where presented and compared. Further discussion of those methods is however out of scope of this thesis.

4.5 K-nearest neighbors

K-nearest neighbors is a popular classification method because it is simple and has a relatively high convergence speed [32]. The idea in which k-nearest neighbors works is that it finds k instances from the training set closest to the instance under analysis. Then, using those instances, the class label is assigned to the analyzed instance. It is done by determining the majority class of the selected training instances. Important in addition to the selected training instances is the distance

metric. It is used to select the closest neighbors. A wide range of different distance metrics, including Euclidean distance, can be used [32]. Important is also the value of K that determines the number of selected neighbors. Small number of neighbors will result in noisy behavior. A too large number will include too many instances in the selected neighbors, resulting in incorrect classifications [19]. There are some things to note when applying the algorithm. If the classes in the dataset to be classified are unevenly distributed, the classifier might not produce good results [33]. If there are a lot of irrelevant or redundant attributes in the dataset, it can affect the classification accuracy of the model. This is why preprocessing the data is important [32].

4.6 Support vector machine

Support vector machines were first introduced in 1992. It is a popular machine learning algorithm that is primarily used for classification, regression analysis and novelty detection. The support vector machine sees training samples as points in space. It is trained by fitting a hyperplane into the space such that the distance between edge instances of classes to be separated is the largest. These points define margins for the said hyperplane and are called support vectors. The training of a support vector machine results in one solution, which is the optimum one. When the support vector machine is trained, it requires that all the data is available and stored in memory. When the training is completed the model only relies on support vectors for classification. New data samples are classified by seeing on which side of the learned hyperplane they fall. Support vector machines provide a regularization mechanism for controlling model complexity. This way it can avoid over- or underfitting. The complexity of classification increases as the amount of used support vectors increases. It does not happen with increasing dimensions of the input space [19].

Support vector machine is capable of classifying dataset items that would otherwise be inseparable. Let us assume that set of datapoints is displayed in one dimensional space. The set contains samples of two different classes. Let us also assume that all the points of one class are surrounded by the samples from the other class from both sides. In this case, separating the classes with a single border is not possible. Support vector machine makes classifying the described data possible with the help of kernel functions. The function introduces a new dimension to the dataset, making the separation of the dataset possible. There are many possible options to use as a kernel function, but the optimal one is usually selected via the means of cross validation [34].

5 Model evaluation with dataset

For this thesis, five different machine learning models were evaluated. The B2B dataset was first processed so that its attribute values were encoded into numerical values with label encoder from Scikit-learn preprocessing library and then one hot encoded with label binarizer from the same library. Readymade implementations of the models found in Scikit-learn python library were used. These models were random forest, decision tree, multi-layer perceptron, k-nearest neighbors and support vector machine.

Random forest was included in the evaluation, because of its positive traits mentioned in the literature review. In Bohanec et al [3], the model was selected because it was broadly recognized as a robust and well-performing machine learning method. There it was also mentioned that black box models achieve generally better results than simpler models. Random forest belongs to this group of black box machine learning models. The study of Bohanec et al [8] evaluated five different machine learning models using a dataset, the attributes of which were based on Bohanec et al [10]. For all the models, AUC (area under curve) and CA (classification accuracy) was calculated. Random forest had the best performance with AUC of 0.84 and CA of 0.776 closely followed by naïve Bayesian classifier with AUC of 0.81 and CA of 0.765. Bohanec at al [8] also included a mention of the robust performance of random forest. Similar mention is also in Meire et al [11].

The multi-layer perceptron is also a black box model and it was considered in Bohanec et al [4], Bohanec et al [8] as a possible alternative machine learning model for the business to business application. In Sharma and Syam [2] it was stated, that feed forward neural networks are most commonly used in business applications, so it was included as an alternative black-box model in the thesis. Support vector machine was also included as an alternative black box model for multilayer perceptron and random forest.

The rest of the models, namely decision tree and k-nearest neighbors were included in the thesis, because they are more simple machine learning models. This was to bring contrast against the three mentioned black box models and to see if simpler models could reach satisfactory results with the B2B dataset. In order to find the best machine learning model out of the presented five, a nested cross validation was run for each of them. In the inner cross-validation of the procedure, possible hyperparameters for the models were tested.

Since each of the selected models were from Scikit-learn, it introduced some restrictions on the possible hyperparameters for each of the models. For support vector machine, available kernels were tested. Four different functions were mentioned in the documentation of the model and

included “linear”,” poly”,” rbf” and “sigmoid”. For multi-layer perceptron, in addition to testing the possible topologies, available activation functions were tested. Scikit-learn provides three functions, which are “logistic”,”tanh” and “relu”. For k-nearest neighbors, values for K were tested from a range of 12 to 200. For decision tree, values for maximum depth were tested from a range from 1 to 50. For support vector machine both the used kernel and value for C were optimized. Scikit-learn offers four kernel options, which are “linear”, “poly”, “rbf” and” sigmoid”. C-values were optimized from a range of exponentially increasing values. Base number for these values was 2 and the exponents ranged from -4 to 6. Ten values were drawn. Finally, the number of estimators for a random forest was tested from a range of 50 to 500.

For the data set analysis, graphs and later for the developed system Python 3.6 with following libraries will be used:

- matplotlib 3.0.3
- pandas 0.24.2
- numpy 1.16.2
- Scikit-learn 0.20.3
- scipy 1.2.1

5.1 Cross-validation

Cross-validation can be used to asses and select a machine learning model [35]. Already in the 1930, it was noticed that if a machine learning model is trained and tested using the same data, the results are overly optimistic. Usually, when conducting cross-validation, there is not that much data available. That is why, the data is split for the validation. One part of the data is used to train the model while the other is used to test the model’s performance [36]. Sometimes the previous domain knowledge can help in selecting the best model for the task. However, when the data is received for the task, the knowledge from it can be inconclusive. That is why Schaffer [37] used cross-validation to select the best model. According to them, using cross validation for model selection can lead to a better average performance while at the same time guarding against catastrophic performance.

Cross validation can be used also to assess the performance of a machine learning model. One option to conduct this assessment would be to use leave-one-out cross-validation. In this type of

cross-validation, only one data instance is used to validate the model while the rest of the dataset is used for training. This method is almost unbiased, and it also reduces the influence of random pairings of data instances from the dataset [38].

In Krstajic et al [35], cross-validation pitfalls were introduced. According to their study, one pitfall would be to report either cross-validation error or single nested cross-validation error as an estimate of the error. Also, according to them, a pitfall would be to select model parameters prior to cross-validation or to select the model based on performance of a single cross-validation iteration.

Varma and Simon [39] state, that if a classifier has been tuned using cross-validation and then afterwards the error estimate is calculated with cross-validation as well, significant bias is introduced in the estimate of the model performance. As a measure to counter the discussed cross-validation bias, the study presents an improved cross-validation procedure. The procedure does not try to present the error estimate of a certain classifier. Instead, a nested cross-validation procedure is used. In the inner cross-validation loop the optimum parameters for a classifier are determined by minimizing the cross-validation error estimate. In the outer loop, the error estimate for this optimized classifier is calculated. It is the almost unbiased estimate of the true error, as stated in Varma and Simon [39].

5.2 Implementation

The nested cross validation was conducted by using a self-implemented algorithm. The algorithm accepts the instances of B2B dataset in a list and the target values of the instances in another list. In addition to these a string parameter that determines the classifier type is passed into the algorithm. First, before the outer part of the procedure, the algorithm instantiates a "**StratifiedKFold()**", with 10 splits. These splits are then extracted with a for-loop. It starts the outer part of the nested cross-validation. The larger split is handed over to the inner cross-validation, whereas the smaller one is saved as a validation set for later. The inner part of the cross-validation and also the optimum parameter search is conducted by a method called "**inner_search()**". It receives a part of the dataset, stratified K-fold instance with nine splits, type of the classifier used and numerical values for use in the optimum parameter search. These values can be either single values in a list or tuples within a list that represent either a range of integer values or a topology of multi-layer perceptron's hidden layers. The training and testing indexes are extracted from the K-fold instance with a for-loop. Within the loop, the type of the classifier is determined. Passed numerical values and possible textual variables, such as activation functions in the case of multi-layer perceptron, are then tested

in different combinations. For each combination, the inner cross-validation is run. Within each iteration of the validation, the selected model is fit to the data, AUC is calculated and saved in a list. Once the inner cross validation has been completed for one combination of parameters, the mean of the saved AUC-values of the inner validation is saved in a list. Also, the parameters that were used during the inner cross-validation are saved in a list. Once the inner cross-validation is finished the list of average AUC-values and the parameters, that caused the highest average AUC, are returned. The returned parameters are then used to instantiate the analyzed classifier in the outer cross-validation. The resulting classifier is fit to the larger part of the dataset extracted in the outer cross-validation and validated with the saved validation data. After this, the AUC is calculated and saved in a list like in the inner cross-validation. Knowledge of the highest AUC is also maintained in the outer part of the validation. If the calculated AUC-value is higher than the one calculated during the previous iteration of the outer validation, the value for highest AUC is updated. The whole process then starts again and is repeated until no splits are left in the stratified k-fold instance of the outer cross-validation. The mean of all the saved AUC-values is the return value of the entire procedures.

5.3 Results

The performance of all the selected models was evaluated with nested cross-validation. The deterministic models, namely k-nearest neighbors and support vector machine were evaluated once. The rest of the models are non-deterministic. Because of this, the evaluation was done four times. It was to see how much variation could be seen in the results. The average AUC-value was calculated from all the tests for all non-deterministic models. This average is used when comparing performance with other models. The optimum parameter search was conducted for all the models, four times for the non-deterministic ones. All the values resulting from the testing were gathered in a spreadsheet. In addition to this a bar-chart was produced to visualize the performance of the tested models.

model	RUN 1	RUN 2	RUN 3	RUN 4	AVG
Random forest	0.826	0.832	0.833	0.832	0.831
Multi layer perceptron	0.755	0.776	0.780	0.788	0.775
Decision tree	0.761	0.753	0.748	0.753	0.754

Figure 2 Nested cross-validation results of non-deterministic models

In the table above there is a collection of AUC-values from the four nested cross-validation runs conducted for the non-deterministic models. The results for random forest are quite stable. The results for decision tree exhibit moderate variation, while multilayer perceptron varies the most. For all those models, the parameter "**random_state**" was set to None. This means that the random number generator used by those models was the random state instance used by "**np.random**". From the values in the table, averages were calculated for each non-deterministic model. They are presented in the following table among the results obtained for the deterministic models.

Classifier	AUC
Support vector machine	0.802
K-nearest neighbors	0.793
Random forest	0.831
Multi-layer perceptron	0.775
Decision tree	0.754

Figure 3 AUC-values obtained from the nested cross-validation

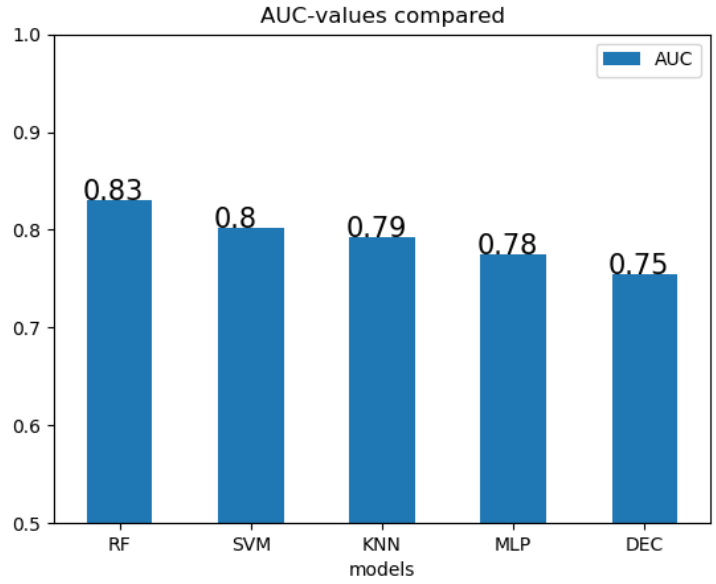

Figure 4 Visualized AUC-values.

In the chart presented above, each bar represents the AUC-value of each tested machine learning model. Each bar is assigned to a model with an abbreviation. “RF” stands for random forest, “MLP” stands for multi-layer perceptron, “DEC” stands for decision tree, “SVM” stands for support vector machine and “KNN” stands for k-nearest neighbors. As can be seen from the bar chart, all the AUC-values are between 0,75 and 0,83.

The worst model out of all the compared models was decision tree with an AUC-value of 0.75. Multilayer perceptron received a slightly better result with AUC-value of 0.78. K-nearest neighbors performed better with AUC-value of 0.79. Support vector machine was the second-best model with AUC-value of 0.80. The best was random forest with AUC-value of 0.83.

Non-deterministic models	Run 1	Run 2	Run 3	Run 4
Random forest (n_estimators)	80	78	143	121
Multi layer perceptron (activation/hidden_layers)	relu/(10,)	tanh/(5,7)	tanh/(10,)	relu/(10,5)
Decision tree (max_depth)	5	8	4	5
Deterministic models	**Param. Search**			
Svm (C/kernel)	29.628/rbf			
K-nearest neighbors (n_neighbors)	86			

Figure 5 Optimal parameters of the tested models

The optimal parameters were searched for each of the tested models after their nested cross-validation. This was done by using the same method responsible for inner cross-validation in the implementation of nested cross-validation. In the leftmost column of the table the tested model is named with a listing of optimized hyperparameters. The parameters are enclosed in brackets and if more than one hyperparameter is optimized, the parameters are separated with a forward slash. Non-deterministic models and deterministic ones are separated. On the top the optimum parameter search results of the non-deterministic models are listed in adjacent columns. The bottom of the chart is reserved for the deterministic models. The optimum parameters are listed in column labeled “Param. Search”.

As can be seen from the table, all the runs of the optimum parameter search for the non-deterministic models produced varying results. This is caused by the nature of the training algorithm all of them have.

As it is discussed in the subchapter of this thesis, that introduces random forests, randomness is involved in growing the decision trees for the forest. At each node of a tree, that is being grown, a variable is selected to split the dataset in two as cleanly as possible. This said variable is selected

from a subset of variables that are in turn selected from all the variables of the dataset at random. So, because of this randomness, sometimes variables that better split the dataset are selected more often to the nodes of grown trees. Because of this, less trees are required to reach the peak predictive performance for the given dataset. On the other hand, if parameters that do not split the dataset well get selected to the nodes of the tree more often, more trees are needed to reach the peak performance.

When it comes to neural networks, the synaptic weights are selected at random before the training. As implied by Bailer-Jones et al [30], different starting weights lead to varying values for the training error, or in the case of this thesis, varying values for AUC. The optimum parameter search, that is conducted within the nested cross-validation, contributes to varying results of the AUC-values as well. Since different starting weights are used each time, different combinations of hyperparameters will produce the optimum results for each iteration of nested-cross validation. This can also be seen from the results of the independent optimum parameter search conducted for multi-layer perceptron.

As it is visible from the produced bar charts and the table displaying the AUC-values obtained for the tested models, the differences between the performances are not huge. But as the results of the nested cross-validation show, random forest performed the best out of all the tested classifiers. It will not however be selected for the system of this thesis. The nested cross-validation and later tests made it clear that the used implementation of random forest performed exceptionally slowly. For this reason, support vector machine will be used.

6 Key features of the system

In this chapter of the thesis the key features and the architecture of the application will be discussed. It aims to test how the requirements stated in the introduction can be implemented. The implementation will be described in the next chapter. In this chapter however the architecture of the application will be described on a general level. Then the key features of the system will be described in depth.

To reiterate the requirements, the application should be robust to concept drifts that can be introduced to the system with the increasing number of sales process instances. Also, the application should be able to automatically produce estimates on how suitable a company would be as a customer for another company without the need of an external consultant.

6.1 Flow of the system

Let us start discussing the implementation of the system on a higher level. When a user takes the system into use, there is necessary not enough data to train a machine learning model for estimating the suitability of a new customer. It still needs to be able to provide recommendations on which customer would be the most beneficial to start a sales process with. Therefore, I suggest a system with two modes. In the first one, no machine learning is used for producing recommendations. The focus of the first mode, besides providing recommendations, is to gather labeled data. The data is then later used as training data for machine learning in the second mode. Depending on the situation, multiple sales processes with a customer can be processed within the system at the same time. Here however, the journey of only one sales process is described. Next the general architecture of the first mode will be introduced.

6.1.1 Mode one

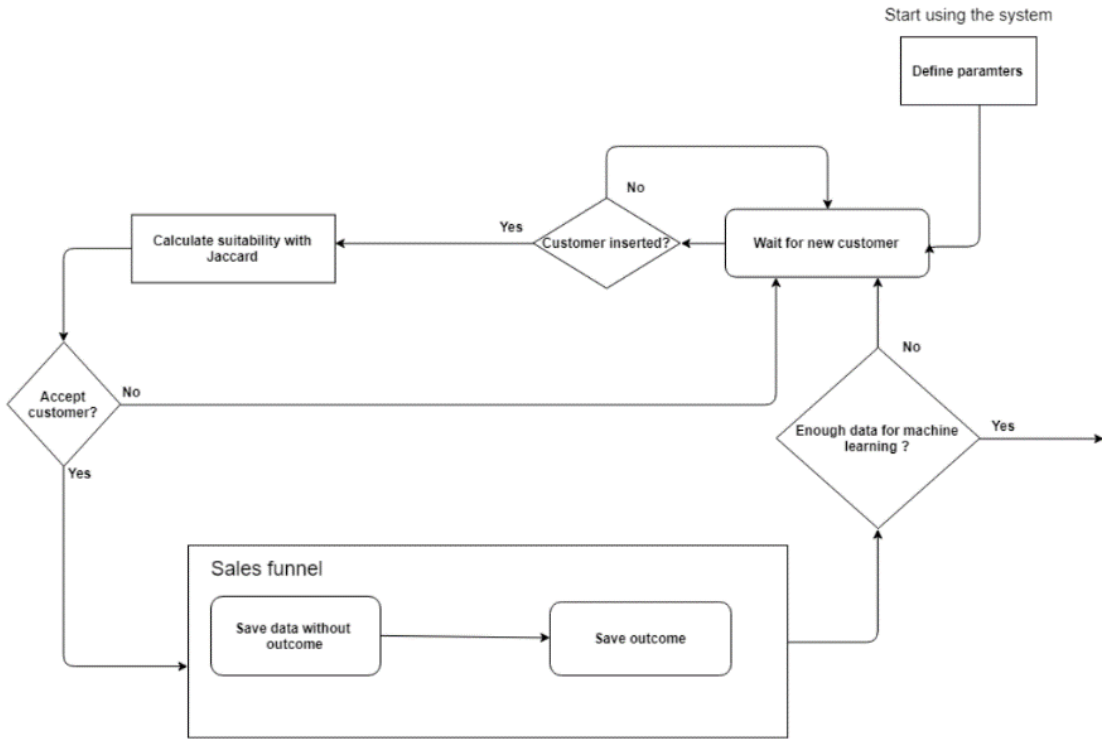

Figure 6 The flow of the first mode

When the user takes the system into use it requires the user to select, which parameters should be gathered from each sales project with a customer company. Here, also the optimal values or ranges should be defined for the selected parameters. The system never returns to this phase automatically. Once the optimum instance is defined, the system is initiated and operates in the first mode. The mode is a loop that starts when a new customer is inserted in the system. Once this is done, the system provides a suitability estimate for the potential customer. The estimation is done by calculating the distance between the saved optimal instance and the potential customer instance. The method of calculating the distance for providing the estimate is discussed later in this chapter. This estimate is offered to the user with a choice to either accept or discard the new customer. This option can potentially introduce bias to the dataset. Since potential customers, that are not deemed as good, can be discarded the dataset might evolve into such state that is not representative of the actual business environment the system operates in. This later can affect the performance of applied machine learning models. If a customer is discarded, the system reverts to phase where new customers are inserted. If a customer is accepted, it is entered in the sales funnel of the company. Sales funnel represents the normal sales process the company uses. Once the sales process with the

new customer is finished, the outcome of the process is saved together with the initially saved customer data. Then the amount of saved data is analyzed. If enough data is saved so that a machine learning model could predict from it, the system moves onto the second mode. If there is not enough data, the system stays on the first mode and waits for a new customer.

6.1.2 Mode two

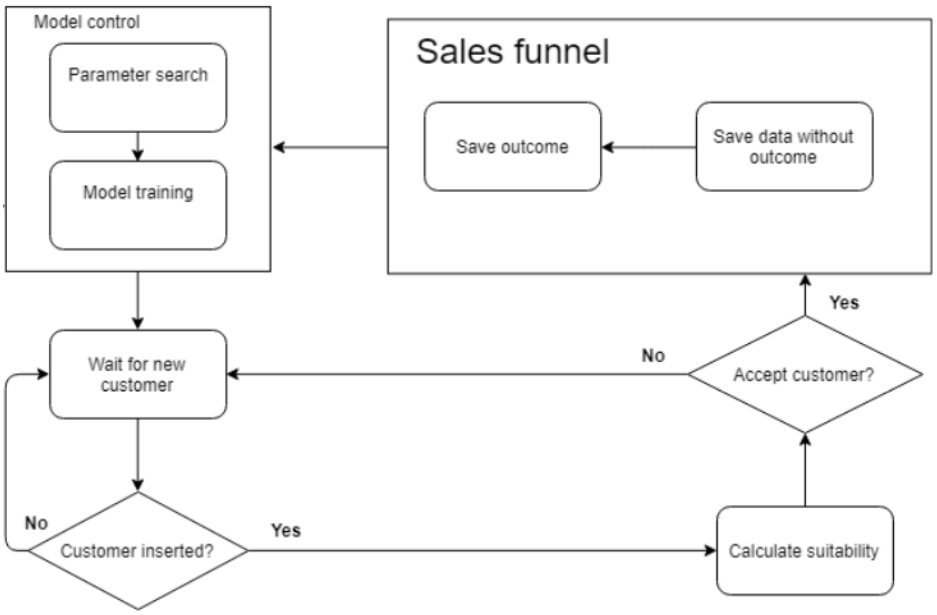

Figure 7 The flow of the second mode

Once second mode is initiated, there is no need for going back to the first one. The mode starts by executing model control part of the mode. The purpose of the module is to search the optimum parameters for the used machine learning model and to train the model with data gathered in the first mode. In the case of this thesis, support vector machine will be tested but depending on the data and preferences of the user, other models can be used. Once the model is trained the general loop of the second mode starts. First, the mode waits for a new customer to be inserted in the system. Once the information of a new customer is inserted, the probability of a sales process with the customer ending as "Won", is calculated. If the customer is accepted based on that, its data is saved. The customer is then moved on to the sales funnel of the company with a sales project. Otherwise the customer is discarded, and the system goes on standby for a new one. This choice offered to the user introduces the possibility of bias like in the first mode. Once the sales project finishes, the outcome of the project is saved before the process exits the sales funnel. Then the

model control is executed again. The dataset has grown by one instance at this point so there is a possibility that drift has been introduced to the concept within the dataset. That is why the model is retrained and selected hyperparameters evaluated to make sure they still are suitable. This is a blind way of dealing with concept drift as stated in Costa et al [14]. The method is not as sophisticated as those introduced in the literature review and with increasing size of the dataset the computational cost of retraining the machine learning model increases. However, the blind way of dealing with concept drift will be enough for this thesis.

6.2 Distance between new lead and optimal parameters.

The first calculation that is executed by the system occurs within the mode one. The distance of the new customer to the optimum one is calculated and used as a suitability measurement for each new instance. In D'Haen and Van den Poel [7] the distance between different instances of data was measured with Jaccard similarity coefficient. It is possible to get this coefficient by calculating the number of variables that belong to the intersection and to the union of the two instances. Once those numbers have been calculated the coefficient is obtained by dividing the number of variables in the intersection with the number instances in the union.

$$S_{jackard}(I_{opt}, I_{new}) = \frac{|I_{opt} \cap I_{new}|}{|I_{opt} \cup I_{new}|}$$

Where I_{opt} represents the user defined optimum instance and I_{new} a new instance the distance of which from the optimum situation is calculated. The values, that are characteristic to a sales project can be mixed. Categorical values, numerical values or ranges of values are possible. The way in which the number of attributes belonging to the intersection is calculated is quite clear: if the values match exactly, the value of the compared variable belongs to the intersection of the two instances. If the compared variables do not match, they belong to the union of the instances. In this thesis a range of values or a list of possible values can be defined for a variable in the optimum instance. If a value of an attribute in a new instance is within the defined range or one of the defined values for the said attribute, it belongs to the intersection.

6.3 Assessing the amount of collected data

For determining when the system can start operating in the mode two, a method is needed to determine if the selected machine learning model is able to predict from the collected data. For this, nested cross-validation will be used. The data gathered in the dataset will be analyzed each time it has grown a certain amount. The resulting AUC-values and their changes will be interpreted as a metric for dataset size. As the gathered AUC-values can vary quite a lot with low amounts of data, a best fit curve will be fitted to the group of calculated AUC-values. If the difference of two latest points of the best fit curve is small enough and above zero, the system will interpret the dataset size to be big enough.

7 Building and testing the system

The system was built as a proof of concept. Target was not to build a ready-to-use system, but to concretize the ideas that would be useful for organizations willing to put together such a system. Each key feature described above namely the function for measuring the distance between the optimum instance and new instance, the feature for analyzing the total amount of instances within the dataset and the model control module will be implemented and demonstrated. The B2B dataset will be used for testing. In this chapter the function of each implemented module will be explained. After the explanations, the function of each module will be demonstrated. To help with the implementation of all the modules, a utility class was implemented for the key features of each module. The demonstrations were extended with three tests. Two of the tests, called A and B, were used to verify the effect of Jaccard filtering to the performance of used machine learning model. The last one was a collective test for the dataset size measurer, for transition from mode one to mode two and for the model control of the second mode. This test was called test C. Each of the parts mentioned will be analyzed in their own sub- chapters. The data for those tests will however be from one run of the system, where each separately described part was run together.

7.1 Offering recommendations with low amount of data

To be able to offer recommendations even without much data, the system offers the possibility to determine an optimal customer. All the potential customers are compared to this instance. The user does this definition, once the system is taken into use. For comparing instances, the Jaccard similarity coefficient described in the previous chapter is used. For the system, a utility class was created that can store the user-defined optimum instance and calculate the Jaccard coefficient. The optimum instance is stored as a python dictionary where the keys are attribute names and values are the possible values of the defined attributes. Values in this dictionary can be exact string values, numbers, ranges of numbers the start and end value of which is defined in a tuple or lists of exact values. For calculating the distance between the new instance and the optimum one, a function named "**calculate_jaccard()**", was defined. It accepts an instance of data that needs to be in a dictionary form. The function keeps track of the number of attributes that belong to the union and intersection of the compared instances. The number of attributes belonging to the intersection and union is calculated within a loop. There all the keys and values of the optimum instance are iterated over. Within each iteration all the possible values are tested. The number of attributes in the intersection and in the union is increased by one if the variable of tested instance:

- matches the value of the same attribute in the optimum instance
- is one of the values defined in a list for the same attribute in the optimum instance
- is a number and within a range defined with a tuple in the optimum instance for the tested attribute
- is any value if the value for the same attribute in the optimum instance is an empty string

The number of attributes in the union is increased by two if the value of the tested attribute in the new instance:

- is not within a range defined in the optimum instance for the attribute
- is not one of the values defined in a list in the optimum instance for the attribute
- does not match the exact value defined in the optimum instance

Once all the attributes have been iterated the method divides the number of attributes in the intersection by the number of attributes in the union and returns the resulting Jaccard similarity coefficient. It is multiplied by 100 to get the value as a percentage.

Since the measurer accepts an instance in a dictionary form, a method for converting an instance into such was created. The method is called "**get_row_as_dict()**". In short, this method accepts the index of the row where the instance is in the dataset and the number of attributes to fetch for it. First the method fetches the desired number of attribute labels from the dataset title row. In the case of the measurer test the number was set to 22. This resulted in the exclusion of the class label. All the other attribute labels were included. Once the labels were fetched, the values that relate to them were fetched from the dataset row determined by the index passed to the method. Finally, labels and fetched values were mapped together and returned in a dictionary. This method was used within another method, called "**get_dataset_categorical()**". As a parameter this method accepts the size of the dataset. Within the method, a list for all dataset instances was defined. It was then returned filled with all the instances of the dataset in the categorial form. Before testing anything, the whole dataset was retrieved in categorical format by using the defined bulk method and in one hot-encoded format to speed up the testing.

In the Jaccard test a loop was defined that set a variable running from zero to the last index of the dataset. This was used to get each instance from the fetched dataset one by one. Each of these instances were fed to the similarity measurer to get the Jaccard similarity coefficient. For the

similarity measurer to work, the optimum instance was defined and passed as a parameter to the constructor of the utility class. Based on the resulting values for the similarity coefficient, the dataset was split in two. The instances receiving a similarity value that exceeded a certain threshold were deemed as suitable instances and the ones that did not get high enough value were deemed unsuitable instances. For both groups, two lists were needed. One was used to store the dataset instances and the other was to store the class labels. The threshold was incorporated in the test to simulate a user that in a real-life situation would make the selection to take the new inserted instance as a customer. For visualization purposes the suitability values were stored in a dictionary so that the keys were the suitability percentages and values the frequencies.

7.2 Testing the similarity measurer

Testing the similarity measurer was started by defining an arbitrary optimal instance. Values for each of the labels in the dataset were set as follows:

- Product: “”
- Seller:””
- Authority: [“Mid”,” High”]
- Comp_size: [“Mid”,”Big”]
- Competitors: “No”
- Purch_dept:”No”
- Partnership: “No”
- Budgt_alloc: “Yes”
- Forml_tend: “No”
- RFI: “Yes”
- RFP: “Yes”
- Growth: “Stable”
- Posit_statm: “”
- Source: [“Joint past”,” Referral”]
- Client: [“Current”,” New”]
- Scope: “Clear”
- Strat_deal: [“Very important”, “Average important”]
- Cross_sale: “”
- Up_sale: “”

- Deal_type: [“Project”,” Maintenance”]
- Needs_def: “Yes”
- Att_t_client: “”

With these values set as the point of reference, the similarity measurer filtered the entire dataset into multiple groups with the suitability percentage as the groups label.

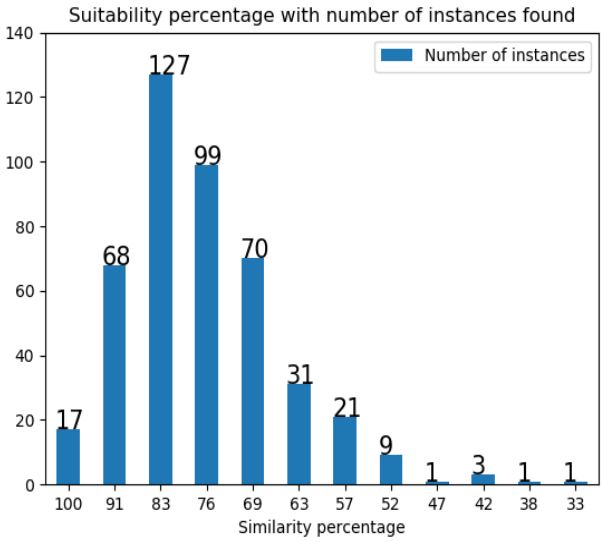

Figure 8 Suitability groups as bar chart

Instances	Suitability
17	100 %
68	91 %
127	83 %
99	76 %
70	69 %
31	63 %
21	57 %
9	52 %
1	47 %
3	42 %
1	38 %
1	33 %

Figure 9 Instances grouped by their suitability percentages

From the illustrations it can be seen, that the implemented similarity measurer can assess new instances based on the optimum instance. Each instance is given a suitability percentage. With it, it

is possible to provide recommendations on how good a customer the new instance would be to the user. Looking back at the defined optimum instance, it would not seem to be that strict. Out of 22 variables in the optimum instance six accept more than one possible value and six accept any value that can be associated with the variable in question. That makes twelve "liberal" variables. The remaining ten have one exact value that needs to be the same in the corresponding attribute of the new instance. This means the slight minority of the variables. With these variables, the system gave suitability of 63% and up for most of the instances in the dataset. Biggest group was 127 instances in size with 83% suitability. Only 17 instances were given 100% suitability. The threshold value in the test was set to 60. It split the dataset into two so that 412 instances were categorized as suitable and 36 as unsuitable.

With these suitability estimates there is one thing that needs to be considered. The optimum instance is defined by the user. Because of this, all the other instances inserted in the system during the mode one are compared to something that represents the user's personal ideas and preferences of a customer. This probably does not represent the real business environment the system is about to be used in. Because of this, the optimum instance should not be decided by just one person. Multiple people should be included in the decision so that the selected optimum values better reflect the real situation. If the decided optimum instance proves to be bad, it is possible to change it. The "bad" optimum instance means that the suitability estimator of the first mode might give better suitability estimates for customers with which a sales process is more likely to end as lost. It should also be mentioned that these "bad" instances could be included in the dataset even with an excellent optimum instance. The estimates are after all there only for guidance.

7.3 Assessing the effect of Jaccard similarity filtering to machine learning

In the introduction of this chapter, tests A and B were mentioned. Here, both were conducted to see what kind of effect the Jaccard filtering would have on the training of a machine learning model. Test A was conducted first. Initialization of the test included defining the same optimum instance for the utility class that was used in the test of Jaccard filter. In addition to this, a list for AUC-lists was defined. The actual tests were run within a loop set to repeat 10 times. After this definition, more preparations followed. They were required by the individual test. First preparation of the individual test A was to shuffle the dataset, that was fetched in one hot-encoded format before any testing.

For shuffling, "**shuffle_data_set()**" method was implemented. As its inputs, it accepts the dataset and class-labels related to the dataset instances. Within the method, a range of numbers from zero to the length of the dataset was gathered in a list. This list was labeled "indexes". The elements of this list were shuffled. Then, two lists for shuffled dataset and class-labels were initialized. They were populated in a loop that iterates over the index list. Each value of the index list was used to determine a position in the dataset from which an instance was selected for the shuffled list. This resulted in a new dataset the instances of which were in random order. List for both shuffled dataset and shuffled class labels was returned.

After shuffling, the dataset and its labels were split into train and test data with stratified train-test-split method of the Sklearn model_selection package. It was used within a method called" **train_test_split_stratified()**" that returns both the train and test data within a dictionary while also taking into account the class distribution of the dataset. With this done, the set of training samples was then iterated over so that each sample was inserted in the dataset of the utility class one by one.

The performance of a machine learning model was then monitored each time two conditions were met. First condition was that at least 100 instances needed to be in the dataset. The second condition was that the dataset size of the utility class had to be 1.15 times bigger than it was when a previous measurement was done. The nested cross-validation implementation was used to conduct the performance measurement. The result of each repetition of the test was a list of AUC-values from the nested cross-validations.

The test B differed in some ways from the test A. Before even adding the first instance to the maintained dataset, the suitability was calculated for the instance with Jaccard similarity. If the suitability was below set threshold, which in the test was 63%, the inspected instance was ignored. Different for this test was also, that it needed the dataset both in categorical and in one hot-encoded format. Because of this, shuffling the both sets and splitting them to train and test data was done here slightly differently. First, a list containing indexes in random order from zero to the length of the dataset was obtained from a separate method. Then, both versions of the dataset were shuffled in random order with a method called "**shuffle_data_set_randomizer()**". This was essentially the same method as "**shuffle_data_set()**", the only difference being that it used the separately generated index list for shuffling. Because of this, both the dataset versions could be shuffled to the same random order. This is important, as after Jaccard similarity was calculated for the instance, the encoded version of it could be added to the dataset of the utility class from the encoded data collection only by using the index known from the categorical list.

Both versions of the dataset were also split to training and testing parts for the test so that 75% of the data is used for training. This was achieved by using "**StratifiedShuffleSplit**" from Scikit-learn model_selection package. With it, training and testing indexes were first generated for the splitting of the both versions of the dataset. They were then used to populate the training and testing partitions for both categorical and one hot-encoded data.

7.4 Analyzing the effect of Jaccard filtering

The effect of Jaccard filtering on the performance of machine learning model was demonstrated by producing two graphs. One depicts how AUC-values develop with instances that have been filtered based on Jaccard similarity and the other depicts how the values develop without any filtering. All the 10 repetitions of both tests were plotted in their own graphs.

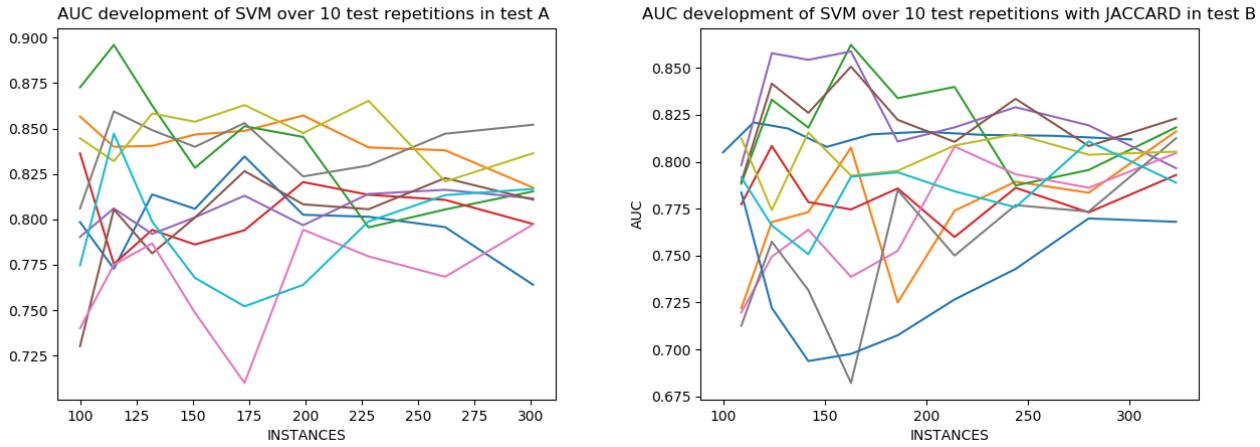

Figure 10 Development of AUC-values in tests A and B

The first test is conducted in all the different repetitions when 100 instances are in the dataset. As it is visible from both charts the initial AUC-values vary quite much. In test A the lowest initial value is below 0.75 and highest is just below 0.875. Initial values are at their densest between 0.775 and 0.85. The spread of initial values in test B is slightly lower than in test A. The curves of test B however spread out quite strongly as the number of instances increase towards 150. The AUC-values of the curves in the region between 100 and 150 instances also seem to vary more than in test A. Highest initial value of B is just above 0.80 and lowest is between 0.70 and 0.725. The denser area of initial values is between 0.775 and 0.825. As the number of instances increase in the dataset the variation of AUC-between the different curves seems to get smaller. In other words, all the curves seem to converge towards a smaller range of values. In both tests the difference of highest

and lowest AUC-value at 300 instances is smaller than at 100 instances. This difference is more noticeable in test B.

In general, these tests highlight well the effect of Jaccard filtering. As the number of instances increase in the dataset during test A, values of all the AUC-value curves converge towards a smaller range of values. This also happens with test B, where the instances are filtered with the Jaccard similarity measurer. Although the AUC-values might vary quite a lot at first with the filtering, over time a more stable AUC can be achieved. In other words, when there is not enough data in the dataset, Jaccard similarity can be used to provide the suitability estimates until machine learning can be used.

7.5 Monitoring the dataset growth

For detecting when it is possible to start using machine learning for providing suitability estimates, a dataset monitoring system was developed. It was tested with support vector machine. For monitoring the size of the dataset, the system uses AUC-values and more specifically a best fit curve that goes through them. Once the difference of two points in this best fit curve is small enough the system determines that there are enough instances in the dataset.

Before testing the dataset control system, the dataset is first fetched in categorical and in one hot encoded format and shuffled in a manner similar to test B. Then both the shuffled sets are split to training and testing data so that 75% of both the sets is used as training data.

The actual test then occurs within a loop that goes through all the instances within the training data. For each instance the Jaccard similarity coefficient is calculated. If this value is high enough, the one hot instance corresponding to the analyzed categorical instance is selected for later addition to the dataset of the utility class. Also, the class label corresponding to this instance is added to a list for suitable class labels.

Before monitoring can take place, there needs to be enough instances stored in the dataset. For this reason, with the inbuilt method of the utility class, the size of the dataset is queried. If the size is less than 100 instances, the only operation done during the iteration is an increment of the dataset of the utility class by one. Once the threshold of 100 instances is exceeded, the dataset is incremented by one as well. In addition to this a new list for class labels is defined and filled with values from the list containing class labels of the suitable instances. The whole list of class labels is not transferred, only from indexes 0 to *index* + 1, where *index* refers to the current value of the iteration.

After this, a new test is conducted. The test controls that there are at least two occurrences of the both class labels in the new class-label list. Also, it is monitored that the size of the dataset has grown by 1.15 from the previous monitoring. These tests are linked with AND-operation. The count of both class labels needs to be checked so that the later cross-validation of the monitor system can give probability estimates for both classes. The growth of the dataset also needs to be monitored. It is to ensure that each time the size of the dataset is monitored, meaningful changes in the results can be observed.

Once all the mentioned tests are passed, the monitor system starts the actual operations needed to assess the development of the dataset. First, the size of the dataset during this iteration is saved in a variable. This is used to determine the next suitable iteration for such control. After this, nested cross-validation is run for the current state of the dataset. Once the AUC-value is obtained from the validation, it is stored in a list.

The actual analysis of the dataset size is started, once the size of the AUC-list is over three. A best fit curve is fit over the stored AUC-values. After that, the steepness of the line is analyzed. The points for the best fit curve are generated by using "**UnivariateSpline()**" from Scipy. The method accepts two lists. One is for the AUC-values and the other is for the number of instances in the dataset each time the AUC was calculated. In turn, it returns a list containing the values for the best fit curve. From these points it is determined when the system should make the transition from the mode one to mode two. Mainly for this, the last two points in the best fit curve are considered. The value of second to last AUC is subtracted from the last. If this difference is close to zero, the dataset has enough data for transition. In the test, the difference values that were considered suitable for transition, were between zero and 0.25.

7.6 Demonstrating the dataset control system

The dataset control system test was executed once with support vector machine. As mentioned in the previous chapter, the code of this test mirrored that of the test B with the dataset size requirement of 100 and a suitability requirement for an instance. For this test it was set to 63%. Each time the size of the dataset was monitored during the tests, the optimum parameter search was run for the used support vector machine. The result of the monitoring system test was a list of AUC-values and a curve depicting the best fitting curve to the stored AUC-values. This sub-chapter begins the collective test C, described in the introduction of this thesis. Next, the produced best fit curve for random forest is displayed and analyzed.

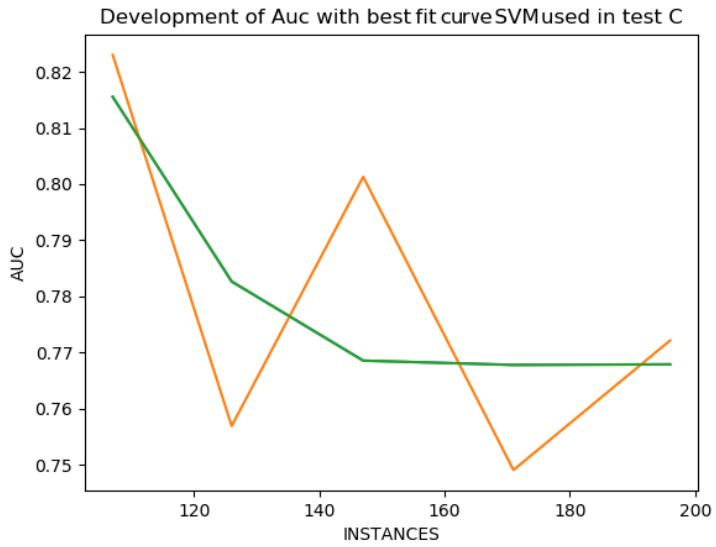

Figure 11 AUC-values with best fit curve marked in green

From the figure it is visible, that the AUC-values plotted in orange vary heavily as the size of the dataset increases. The best fit curve plotted in green, changes much less and is thus more suitable for the dataset analysis. When there were 180 instances in the dataset, the dataset monitoring system triggered a transition from mode one to mode two.

7.7 Model control of the second mode

Once there is enough data in the dataset, the system transitions to mode two. The most important part there, is the model control module. For this thesis, it was simulated by repeatedly fitting selected model to the data within the gathered dataset. Then, instances from test data list were added one by one to the dataset to see if there is significant change in the performance of the used machine learning model. The test data mentioned here is from the same dataset split the training data of which was used in chapter 7.5. Also, the way in which the suitability estimates were calculated in the second mode was tested here. Once a classifier has been fitted with data it is possible to calculate probability estimates for each of the available classes when a new data instance is given to the classifier. The probability of the new instance being classified as “Won” is used here as the suitability measure for a company.

Before starting the model control system simulation, the suitable class instances are transferred to a general list labeled “classes”. Then three lists are instantiated, one for AUC-values that are again used to measure the performance of the machine learning model, one for storing number of

instances in the dataset during each of the measurements and dictionary for calculated probability estimates. The actual simulation then occurs within a loop, that iterates through a range of numbers from zero to length of the test instance list. First operation within the loop is to conduct optimum parameter search for the used machine learning model. For this, the inner validation method described earlier in this thesis is used. Then, a classifier is initialized with the obtained optimum parameters and fit to the current state of the dataset. After this, an instance is taken from the one hot encoded test data. With the classifier, the probability estimates are calculated for the instance. The one for the "Won"- class is multiplied by 100 after this. This value in real application is the customer suitability value. Following this the suitability percentage is stored in a dictionary with its frequency as the key. This is not necessary in a real situation but is used here to enable the visualization of all different suitability percentages found within the test data. After this, the performance of machine learning model is measured with nested cross-validation. The resulting AUC-value is stored in a list. The test iteration is finished by storing the analyzed instance within the dataset of the utility class and the corresponding class label is stored within the classes list. This continues until all the instances in the test data are processed.

7.8 Testing the model control of the second mode

The model control module was tested after the system transitioned from mode one to the mode two. AUC-values and probability estimates were gathered during the test. AUC-was used to see the development of the support vector machine performance as the dataset size increased. The calculated probabilities were used on the other hand to visualize the suitability distribution found by the classifier. Probability for the "Won" class was used as the suitability. This sub-chapter continues the test C. From the test run, following suitability percentages were gathered.

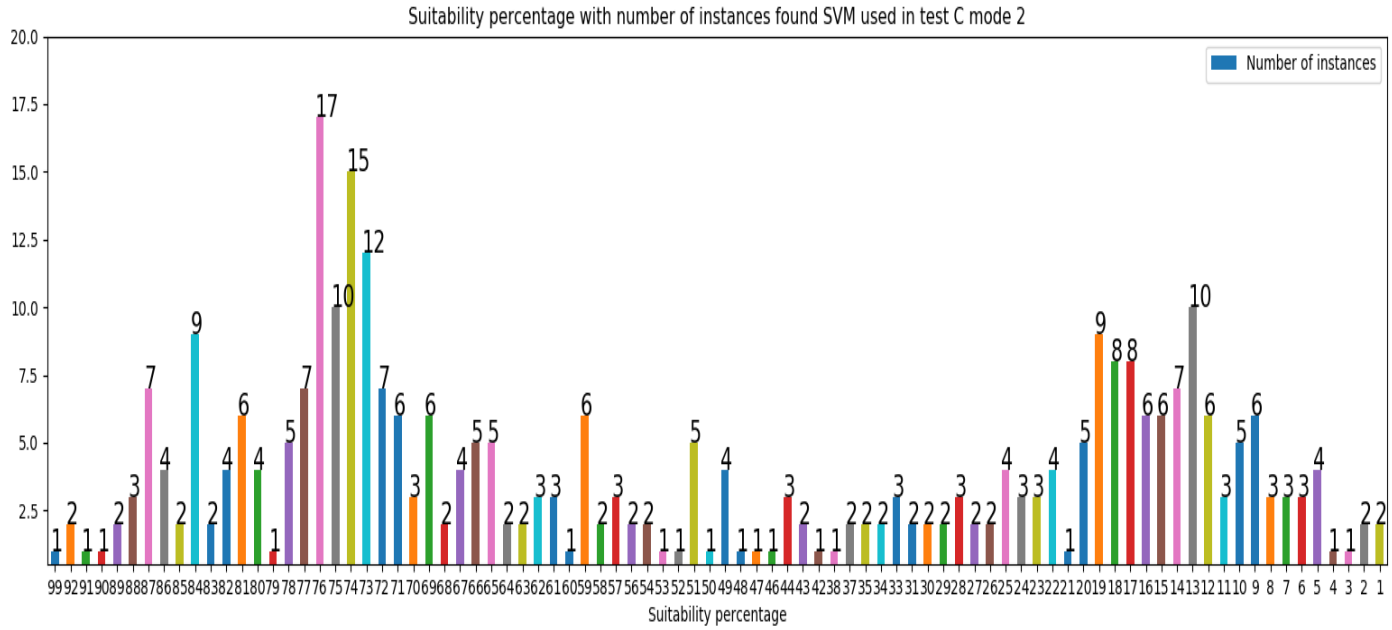

Figure 12 Suitability percentages obtained with support vector machine

The first observation from the visualized suitability percentages would be that the used classifier found a more fine-grained selection of different suitability percentages than the Jaccard similarity filter. The best suitability percentage was 99 and worst was 1. The frequencies of the found suitability percentages vary between 1 and 17. In my opinion the results visualized in the bar chart look realistic. For each instance, the suitability percentage was calculated from a part of the dataset the classifier has never seen. For this reason, all the found suitability percentages are distributed the way they are. In the chart, there are two clusters of instances that stand out. One is between 70% and 80% and the other one is between 5% and 20%. The bar chart would look different if the classifier were to analyze instances it already knows. For example, the mentioned cluster between 70% and 80% would be closer to the better end of the suitability spectrum and in the cluster, there would be more instances.

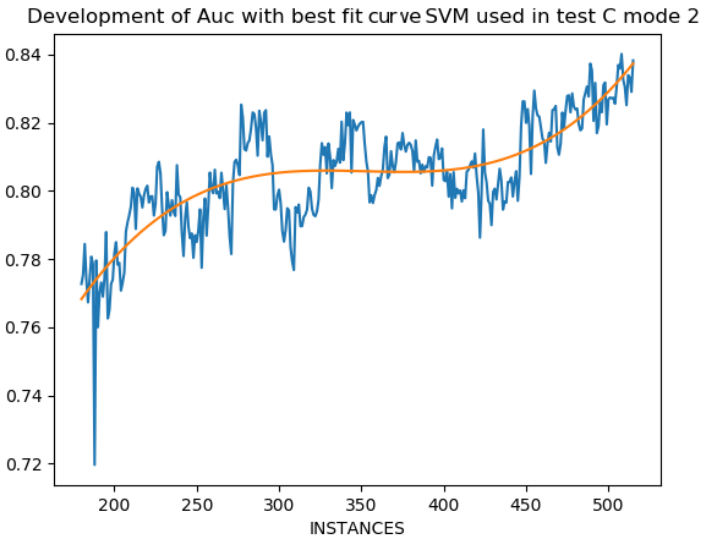

Figure 13 AUC development of k-nearest neighbors

During the test, the AUC-values were also calculated after each added new instance to the dataset. For that, the nested cross-validation implementation was used. The general idea of the test was to see how increasing the dataset with new instances affects the performance of the model. As it can be seen from the AUC-value curve the value changes considerably between the added instances. The orange best fit curve however would indicate that the general direction of the AUC-values is towards the better ones. This would suggest that adding new instances would not have too adverse effects on the model performance. With different data instances or different order of the added instances, there is however potential for different looking plot.

8 Conclusions and further work

As a part of this thesis, a demonstrative system for analyzing customer suitability was developed. It had two different modes for operation. The first of them did not utilize any machine learning for providing estimates of suitability while the other one did. All the different modules of the system were introduced separately. The similarity measurer of the first mode was demonstrated separately from the other modules. The test results of the rest of the modules were analyzed in their own chapters. Data for those chapters was produced from a program execution where all the modules were integrated into a complete system. The aim in building the system was to test the ideas needed to build such a system rather than to build a ready to use implementation. If a company is willing to put such a system together, the ideas tested in this thesis could be used.

This thesis had three research questions. First one asked, how can machine learning be used to provide suitability estimates for potential customers. The problem with using machine learning was the availability of data. Without data, providing estimates would not be possible with machine learning. Through research, an open dataset was found that listed sales processes and their outcomes. It was mentioned in Bohanec et al [3]. In that study, machine learning was used to forecast the outcomes of sales processes. This thesis however used the sales process data to measure the suitability of a potential customer. First as the user begins to use the system there potentially is no data gathered in the system for them. The found dataset was used to simulate the flow of new customers to the system. First with low number of instances, the suitability estimates were provided only with Jaccard similarity coefficient. Machine learning was only used to measure the size of the gathered dataset. As the size of the set grew big enough, the system changed modes and started to offer recommendations with machine learning. The gathered data was used as training data in mode two. Once a completely new instance was inserted to the system operating in that mode, instead of predicting its outcome, the probability estimates for both outcomes were calculated. The probability estimate for the “Won”-class was used as the suitability estimate for the new customer.

The second research question asked how much can be automized from creation and use of the machine learning models in customer suitability estimation. In Bohanec et al [4] the machine learning model was used by an external consultant and the results of machine learning predictions were analyzed later in a meeting. For comparing different machine learning models and finding out their optimum parameters, code was developed for this thesis. Nested cross-validation was used for comparison, and the inner part of the validation was used for the optimum parameter search. So, if a

company wants to adopt this system into their processes, a selection of machine learning models could be compared with said methods to determine the best one.

The question of how the use of machine learning models can be automatized is answered with the developed code of this thesis. For analyzing the size of gathered data in the mode one, machine learning was used. There, nested cross-validation with selected model was integrated in the system to see when to move onto next mode. The resulting AUC-values were gathered from the validations and a best fit curve was used to determine the transition point. In the second mode, model control module included machine learning. The used machine learning model was retrained each time a new instance was added to the system and then used to provide the suitability estimate.

The final research question was answered by first researching concept drift and possible ways of detecting and dealing with it. In Costa et al [14] blind way of dealing with concept drift was mentioned. Essentially it means retraining the used machine learning model each time a new instance is added to the dataset. Since other methods seemed quite complex the blind way was implemented in the constructed system. The training occurring in model control of mode two, is exactly for dealing with the concept drift.

The gained speed by using the developed modules of this thesis within a decision support system are speculations. Nevertheless, some of the possible benefits are as follows: instead of having to compare multiple different company customer candidates manually the system allows the user to define what he or she considers the best characteristics for their customers. Based on this the system does the comparison work for the user while saving time. There is however one time-consuming part still in the system, which is putting in the data of a new and potential customer. The data still needs to be inserted manually and searching for different parameters from different sources can take a lot of time. The system however could be extended so that all the required data is fetched from an external information provider via an API. Selecting the most suitable API is best left for each user's own consideration. Using this kind of API would reduce the users work to just making the decision on which customer to accept as a new customer. The system is also capable of adapting to changing business environments. Although it was mentioned previously in this thesis that the system never automatically returns to the definition of optimum instance in mode one, it is not prevented. If the user notices that the optimum instance needs to be changed, it is possible. In mode two the used machine learning model is retrained each time a new instance is added to the dataset. This way, the concept drift possibly introduced by the new instance is dealt with and the model used provides accurate enough estimates.

Once the internal logic is implemented with such a way that is suitable for the user and their company, the presentation of customer candidates and their suitability percentage should be thought of. I suggest that it should not be too complex. When a user is inserting data, the suitability could be shown as soon as the user has filled in all the information of the new company. For inserting the data, a simple form could be implemented, and the suitability could be shown within the form as a percentage. The percentage could be situated near buttons that allow accepting or rejecting the customer candidate. This would be a possibility with a version of the system that does not have an external data-provider integrated.

With the version, that has some sort of external data-provider integrated a module could be added that searches potential customers based on inserted search criteria. These customers could be then presented in a list. The item of this list does not have to have all the information visible from the company. The name along with the suitability could be enough. A nice visual addition would be for example a colored dot that can have a range of colors going from green via yellow to red. Green would obviously mean the most suitable and red the least suitable instances. By clicking such item, the company form with complete information could be opened.

In conclusion, it can be stated that by careful planning it is possible to use machine learning to assess the suitability of a potential customer. The modules of this thesis integrated in a system suitable for a certain company will not offer absolute truths. However, they can be used as a decision support when analyzing which companies are the most beneficial to be taken as a customer. Although it still is recommended to analyze the gathered dataset while the system is in use the need for external participants as in Bohanec et al [3] is no more.

This thesis considered aspects that would be relevant when developing a program for producing recommendations, rather than considering paradigms for using machine learning within a company as an internal process. Only in D'Haen and Van den Poel [7] of all the used sources of this thesis, a programmatical tool was developed for utilizing machine learning. It inspired the proposed two mode approach with dataset size monitoring and possibility to start with no gathered data. A way to move forward with the ideas of this thesis would be to start for example an open source project around them. With it, the project would be open to an abundance of ideas and a way towards a better customer suitability analysis tool would be possible.

9 References

[1] “The B2B knowledge gap”, Gary L. Lilien, International Journal of Research in Marketing, 2016, Vol. 33, issue 3, pp 543–556

[2] “Waiting for a sales renaissance in the fourth industrial revolution: Machine learning and artificial intelligence in sales research and practice”, (Arun Sharma, Niladri Syam), Industrial Marketing Management, 2018, Vol. 69, pp 135-146

[3] “Explaining machine learning models in sales predictions”, (Marko Bohanec, Mirjana Kljajić Borštnar, Marko Robnik-Šikonja), Expert Systems with Applications, 2017, Vol. 71, pp 416–428,

[4] “Organizational Learning Supported by Machine Learning Models Coupled with General Explanation Methods: A Case of B2B Sales Forecasting”, (Marko Bohanec, Mirjana Kljajić Borštnar, Marko Robnik-Šikonja), Organizacija, 2017, Vol. 50, No. 3, pp 217 – 233

[5] “Sales forecasting by combining clustering and machine-learning techniques for computer retailing”, (I-Fei Chen, Chi-Jie Lu), Neural Computing & Applications, 2017, Vol. 28, issue 9, pp 2633-2647

[6] “Strategic Foresight of Future B2B Customer Opportunities through Machine Learning”, (Daniel Gentner, Birgit Stelzer, Bujar Ramosaj, Leo Brecht), Technology Innovation Management Review, 2018, Vol. 8, issue 10, pp 5 – 17

[7] “Model-supported business-to-business prospect prediction based on an iterative customer acquisition framework”, (Jeroen D'Haen, Dirk Van den Poel), Industrial Marketing Management, 2013, Vol. 42, Issue 4, p 544 – 551

[8] “Decision-making framework with double-loop learning through interpretable black-box machine learning models”, (Marko Bohanec, Mirjana Kljajić Borštnar, Marko Robnik-Šikonja), Industrial Management & Data Systems, 2017, Vol. 117, Issue 7, pp 1389-1406

[9] “Predicting customer profitability during acquisition: Finding the optimal combination of data source and data mining technique”, (Jeroen D'Haen, Dirk Van Den Poel, Dirk Thorleuchter), Expert Systems with Applications, 2013, Vol. 40, Issue 6, pp 2007 – 2012

[10] “Modeling attributes for forecasting B2B opportunities acquisition”, (Marko Bohanec, Mirjana Kljajić Borštnar, Marko Robnik-Šikonja), 2015

[11] “The added value of social media data in B2B customer acquisition systems: A real-life experiment”, (Matthijs Meire, Michael Ballings, Dirk Van Den Poel), Decision Support Systems, 2017, Vol. 104, p 26–37

[12] Aditee Jadhav, Leena Deshpande: An Efficient approach to detect concept drifts in data streams. In: 7th International Advance Computing Conference (Hyderabad, India), 2017

[13] Bruno I. F. Maciel, Silas G. T. C. Santos, Roberto S. M. Barros: A lightweight concept drift detection ensemble. In: 27th International Conference on Tools with Artificial Intelligence (Vietri sul Mare, Italy), 2015

[14] Albert França Josuá Costa, Régis Antônio Saraiva Albuquerque, Eulanda Miranda dos Santos: A Drift Detection Method Based on Active Learning. In: International Joint Conference on Neural Networks (Rio de Janeiro), 2018

[15] B2B Dataset: http://www.salvirt.com/research/B2Bdataset/

[16] “How many trees in a random forest?” Lecture Notes in Computer Science, (Mayumi Oshiro, Thais & Santoro Perez, Pedro & Baranauskas, José), 2012, Vol. 7376, p 154-168

[17] Robert Reris, J.Paul Brooks: Principal Component Analysis and Optimization: A Tutorial. In: 14th INFORMS Computing Society Conference Richmond, Virginia, 2015, January 11-13, pp. 212-225

[18] “Introduction to Principal Components Analysis”, (Paul J. Francis, Beverly J. Wills), *arXiv.org*, 1999

[19] “Efficient learning machines. Theories, Concepts, and Applications for Engineers and System Designers.”, (Mariette Awad and Rahul Khanna), (2015), p 1-15, p 24 (Random Forest), p 30-32 (Principal component analysis), p 39 – 43 (Support vector machine), p 129, 133 (Classical ANN)

[20] “A very brief introduction to machine learning with applications to communication Systems”, IEEE Transactions on Cognitive Communications and Networking, (Oswaldo simeone), 2018, Vol. 4, issue 4, p 648-664

[21] “Overfitting and Undercomputing in Machine Learning”, (Tom Dietterich), ACM Computing Surveys, 1995, Vol. 27, issue 3, p 326 – 327

[22] “Unsupervised learning: Foundations of neural computation A Review”, (DeLiang Wang), AI Magazine, 2001, Vol. 22, issue 2, p 101

[23] “Semiboost: Boosting for semi-supervised learning”, (Pavan Kumar Mallapragada, Anil K. Jain, Rong Jin, Yi Liu), IEEE Transactions on pattern analysis and machine intelligence, 2009, Vol. 31, issue 11, pp 2000 – 2014

[24] “Marketing Segmentation Through Machine Learning Models an Approach Based on Customer Relationship Management and Customer Profitability Accounting”, (Raquel Florez Lopez, Juan Manuel Ramon-Jeronimo), Social Science Computer Review, 2009, Vol. 27, issue 1, pp 96 – 117

[25] “What are decision trees?”, (Carl Kingsford, Steven L Salzberg), Nature Biotechnology, 2008, Vol. 26, issue 9

[26] “Random forests”, (Leo Breiman), Machine Learning, 2001, Vol. 45, issue 1, p 5-32

[27] “Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500”, (Cristopher Krauss, Xuan Anh Do, Nicolas Huck), European Journal of Operational Research, 2017, Vol. 259, Issue 2, p 689 – 702,

[28] “Artificial neural network models for forecasting and decision making”, (Tim Hill, Leorey Marquez, Marcus O’Connor, William Remus), International Journal of Forecasting, 1994, Vol 10, Issue 1, p 5-15

[29] “A review paper on Artificial Neural Networks”, (Rani Pagariya, Mahip Bartere), International Journal of Advanced Research in Computer Science, 2013, Vol. 4, issue 6

[30] “An introduction to artificial neural networks”, (Coryn A. L. Bailer-Jones, Ranjan Gupta, Harinder P. Singh), 2001, Automated Data Analysis in Astronomy, arXiv:astro-ph/0102224

[31] Tijana Vujičić, Tripo Matijević, Jelena Ljucović, Adis Balota, Zoran Ševarac:“Comparative Analysis of Methods for Determining Number of Hidden Neurons in Artificial Neural Network”. In: Central European Conference on Information and Intelligent Systems, 2016, p 219 – 223

[32] Mashael S. Aldayel: “K-Nearest Neighbor Classification for Glass Identification Problem”. In: International Conference on Computer Systems and Industrial Informatics, Sharjah, 2012, p. 1-5,

[33] Shiliang Sun, Rongqing Huang: “An Adaptive k-Nearest Neighbor Algorithm”. In: Seventh International Conference on Fuzzy Systems and Knowledge Discovery, Yantai, 2010, p 91-94

[34] “What is a support vector machine?”, William S. Noble, Nature Biotechnology, 2006, Vol. 24, issue 12, p.1565

[35] “Cross-validation pitfalls when selecting and assessing regression and classification models”, (Damjan Krstajic, Ljubomir J Buturovic, David E Leahy, Simon Thomas), Journal of Cheminformatics, 2014, Vol. 6, issue 1, pp 1 -15

[36] “Survey of cross validation procedures for model selection”, (Sylvain Arlot, Alain Celisse), Statistical surveys, 2009, Vol.4, pp 40 – 79

[37] “Selecting a classification method by cross validation”, Cullen Shaffer, Machine learning, 1993, Vol. 13, issue 1, p 135 - 143

[38] “An Efficient Leave-One-Out Cross-Validation-Based Extreme Learning Machine with Minimal User Intervention”, (Shifei Shao, Meng Joo Er, Ning Wang), IEEE Transactions on Cybernetics, 2016, Vol 46, issue 8, pp. 1939-1951

[39] “Bias in error estimation when using cross-validation for model selection”, (Sudhir Varma, Richard Simon), BMC Bioinformatics, 2006, Vol. 7