UNIVERSITY OF TURKU

ABSTRACT

	Bachelor's thesis
x	Master's thesis
	Licentiate's thesis
	Doctor's thesis

Subject	Information Systems Science	Date	30.4.2019
Author(s)	Saila Collan	Student number	51112
		Number of pages	83 + appendix
Title	BENCHMARKING CLASSIFIERS - HOW WELL DOES A GOWA-VARIANT OF THE SIMILARITY CLASSIFIER DO IN COMPARISON WITH SELECTED CLASSIFIERS?		
Supervisor(s)	Ph.D. Jani Koskinen Prof. Pasi Luukka		

Digital data is ubiquitous in nearly all modern businesses. Organizations have more data available, in various formats, than ever before. Machine learning algorithms and predictive analytics utilize the knowledge contained in that data, in order to help the business related decision-making. This study explores predictive analytics by comparing different classification methods – the main interest being in the Generalize Ordered Weighted Average (GOWA)-variant of the similarity classifier.

The target for this research is to find out how what is the GOWA-variant of the similarity classifier and how well it performs compared to other selected classifiers. This study also tries to investigate whether the GOWA-variant of the similarity classifier is a sufficient method to be used in the business related decision-making. Four different classical classifiers were selected as reference classifiers on the basis of their common usage in machine learning research, and on their availability in the Statistics and Machine Learning Toolbox in MATLAB.

Three different data sets from UCI Machine Learning repository were used for benchmarking the classifiers. The benchmarking process uses fitness function instead of pure classification accuracy to determine the performance of the classifiers. Fitness function combines several measurement criteria into a one common value. With one data set, the GOWA-variant of the similarity classifier performed the best. One of the data sets contains credit card client data. It was more complex than the other two data sets and contains clearly business related data. The GOWA-variant performed also well with this data set. Therefore it can be claimed that the GOWA-variant of the similarity classifier is a viable option to be used also for solving business related problems.

Key words	Similarity classifier, benchmarking, classification, machine learning
Further information	

TURUN YLIOPISTO

TIIVISTELMÄ

	Kandidaatintutkielma
x	Pro gradu -tutkielma
	Lisensiaatintutkielma
	Väitöskirja

Oppiaine	Tietojärjestelmätiede	Päivämäärä	30.4.2019
Tekijä(t)	Saila Collan	Matrikkelinumero	51112
		Sivumäärä	83 + liitteet
Otsikko	Luokittimien suorituskyvyn mittaaminen – Miten hyvin similaarisuusluokittimen GOWA-variantti pärjää vertailussa muihin valittuihin luokittimiin?		
Ohjaaja(t)	Ph.D. Jani Koskinen Prof. Pasi Luukka		

Digitaalinen data on kaikkialla läsnä suurimmassa osaa tämän päivän yrityksiä. Organisaatioilla on saatavillaan enemmän dataa, monissa eri muodoissa, kuin koskaan aikaisemmin. Koneoppiminen ja ennustava analytiikka hyödyntävät näiden kyseisten datojen sisältämää tietoa ja ovat apuna liiketoimintaan liittyvässä päätöksenteossa. Tässä tutkimuksessa käsitellään ennustavaa analytiikkaa vertaamalla erilaisia luokitusmenetelmiä - pääasiallinen kiinnostuksen kohde on similaarisuusluokittimessa, joka käyttää GOWA (Generalized Ordered Weighted Average) operaattoria aggregaattina.

Tutkimuksen tavoitteena on selvittää millainen luokitin on similaarisuuteen perustuva GOWA-variantti ja kuinka hyvin se toimii verrattuna muihin valittuihin luokittimiin. Tässä tutkimuksessa pyritään myös selvittämään onko GOWA-variantti similaarisuusluokittimesta sopiva menetelmä käytettäväksi liiketoimintaan liittyvässä päätöksenteossa. Referenssiluokittimiksi valittiin neljä erilaista klassista luokitinta. Valinta pohjautui niiden yleisyyteen koneoppimiseen liittyvässä tutkimuksessa ja niiden saatavuuteen Matlabin ”Statistics and Machine Learning Toolbox” kirjastosta.

Luokittimien vertailuun käytettiin kolmea erilaista data settiä UCI Machine Learning -tietovarastosta. Suorituskyvyn mittaamisessa käytetään pelkän luokittelutarkkuuden sijaan fitness funktiota. Fitness funktio yhdistää useita mittauskriteerejä yhdeksi arvoksi. Yhden käsitellyn data setin kanssa GOWA-variantti suoriutui parhaiten. Yksi dataseteistä sisältää selvästi liiketoiminaan liittyviä tietoja, ollen myös muita settejä monimutkaisempi suuremmalla havaintojen ja attribuuttien määrällä. GOWA-variantti toimi hyvin myös tällä data setillä. Täten voidaan väittää, että similaarisuusluokittimen GOWA-varianttia voidaan käyttää myös liiketoimintaan liittyvässä ongelmanratkaisussa.

Asiasanat	similaarisuusluokitin, suorituskyvin mittaaminen, luokittelu, koneoppiminen
Muita tietoja	

UNIVERSITY OF TURKU
Turku School of Economics

BENCHMARKING CLASSIFIERS - HOW WELL DOES A GOWA-VARIANT OF THE SIMILARITY CLASSIFIER DO IN COMPARISON WITH SELECTED CLASSIFIERS?

Master´s Thesis
in Information Systems Science

Author:
Saila Collan

Supervisors:
Ph.D. Jani Koskinen
Prof. Pasi Luukka

30.4.2019
Turku

The originality of this thesis has been checked in accordance with the University of Turku quality assurance system using the Turnitin OriginalityCheck service.

Table of Contents

List of figures

List of tables

1 INTRODUCTION

1.1 Motivation for the study

As we live in the modern word ruled by data, enterprises operate quite many of their functions in a digital space. Many business functions like marketing, sales, supply chains, transportation, finance, customer relationships, vendor data management etc., all have data assembled and accessible in a ways that never has been before. The challenge is to transform the collected data into a meaningful and usable format and make beneficial decisions based on that data. Business analytics is a field of study that relates to data collection and processing and analyzing the data.

Business analytics is typically an iterative and systematic investigation of some part of organizations data. It aims to create insight about the descriptive and the prescriptive information of the data. Business analytics can also be connected to data that comes from the outside of an organization, but that is relevant to the organization. By using methods of business analytics, including statistical analysis and a number of algorithm-based analytics methods, companies can create models that will help them to make better business decisions or to automate and optimize business processes. (Rouse, 2017). Such decisions could be for instance setting a product price, selecting right target for a marketing campaign, making investment decisions, doing credit scoring or bond classification, predicting churn rate, optimizing supply chain etc. There are a wide variety of alternatives where business analytics can be used for creating a competitive edge.

The focus for business analytics is to create additional value from the data by using various technologies to create applications which analyze the data. These applications aim to produce new insights by extracting useful patterns from the data. The goal for the data-analytic thinking is to understand the core business better and to help the business decision making process. Compared to pure data analytics or data science, in business analytics one needs to have a good understanding about the business context. A good domain knowledge is required to be able to assess the requirements which comes from organization's operations and functions and to be able apply them in the analytic applications. (Provost & Fawcett, 2013).

Even though business analytics resembles data science, there is still a clear distinction between these two. Data science means more custom coding and open-ended question, while practitioners of business analytics use business analytical tools that can perform advanced statistical algorithms automatically, thus requiring less special skills involved in data science. (Rouse, 2017). This thesis relates to business analytics and more precisely to predictive analytics. An analytical tool, MATLAB, is use to resolve classification problems in this thesis.

In a modern business environment, data and analytics are playing a greater role than ever before. It is not that easy anymore to create a competitive edge in traditional ways like differentiation. For instance product or technological differentiation is quite hard to achieve, geographical advantage is pretty much gone in global business etc. But one thing that is still left as a basis of competition is an ability to execute business with utmost effectiveness and to make best business decisions possible. Good decisions are typically based on systematically assembled data and analysis. (Davenport & Harris, 2007).

Bonabeau (2003) notes that while intuition plays an import role in decision making, decisions which are based on analytics are more likely to be correct than those based on intuition. More and more companies are using some kind of analytics to enhance their business processes and analytical skills are without a doubt something that will be useful in the future. This was one of the motivational reasons why business analytics was chosen as a basis for this thesis.

But data as such is nothing new; enterprises have been dealing with their business related data for a long time. Predictive analytics, which uses historical data, statistical algorithms and machine learning techniques to assess the probability of future events, is nothing new either. But with increased and cheaper computing power, better and easily accessible software and increased volumes of valuable data has made predictive analytics one of the technologies of this decade. As Eric Siegel (2013) says: “Predictive analytics unleashes the power of data”. Siegel presents in his book various real life examples of how we are surrounded with predictive analytics in a modern digital world. Predictive analytics drives millions of decisions in healthcare, insurance companies, financial institutions, sales, marketing etc.

Enormous amount of data is constantly generated and gathered in numerous domains with different kinds of digital devices all over the world. Qiu et al. (2016) claim in their survey that digital information has grown nine times in volume in five years in year 2011 and the amount of data will reach 35 trillion gigabytes by the year 2020.

They are several companies, such as IBM, Netflix, Facebook, Google, Amazon, PayPal, BBC etc., which are excellent examples of companies who have beneficially utilized predictive analytics in their business. They have gathered a vast amount of data and then used it to build models to predict their customer behavior. Predictive analytics means that computers will truly learn from the data how they can calculate the future behavior of the individuals or events. It should be evident that perfect predictions are impossible, but it can give you better odds to make successful decisions and when doing that at scale, it will make a difference in the end. (Tao, 2018).

There are many business related application areas that can take advantage from predictive analytics and machine learning applications. Forbes Technology council (2018) listed a few possible business applications for artificial intelligence and machine learning such as cyber-security defense, recruiting automations, health care diagnostics,

reducing energy cost, and usage, becoming more customer centric, market predictions, advanced billing rules including credit scoring to name a few. Companies can benefit from machine learning algorithms by using them to identify trends in massive amounts of data and to make quicker decisions, which can possibly give them a competitive edge. The aim of these analytical models is to produce reliable and repeatable results and to uncover patterns and trends in data that cannot be noted otherwise and doing that at scale.

One part of machine learning that is commonly called "pattern recognition" is based on classification. Classification is an attempt, or the process of matching items of data (typically multiple criteria) to pre-determined classes. The classes are determined typically based on identified data, where the correct class of each piece of data is known. This "labeled data" is used to train classification algorithms so that they are later able to perform classification on some previously unseen data reliably and accurately. (Provost & Fawcett, 2013). You might want to know whether an email is a spam, or not, or to distinguish between legitimate and fraudulent transactions. There are quite many application areas for classifiers in business related problems and decision making. That is why classifiers among other machine learning algorithms were selected as a topic for this thesis. The focus of this research is to study how well different selected classification algorithms compare against each other and how well they work with business data.

The starting point for this research was a research paper by Kurama et al. (2017) which introduces a GOWA (Generalized Ordered Weighted Average) variant of the similarity classifier. The paper shows that the GOWA-variant works quite well, but the paper does not compare the said classifier to other classifiers. This is what is attempted here. The other goal for this research is to investigate whether the similarity-based classifiers are effective in business analytics problem solving. Kurama et al. (2017) use only medical data in their research, but this study will use in addition to that medical data also more business related data, credit card client data, for benchmarking.

1.2 Research focus

This research will concentrate on studying and benchmarking the GOWA-variant of the similarity classifier. The main empirical contribution of the work will focus on testing the performance of the said classifier against a set of selected classifiers. The closer study of the many other possible classifiers (that do not belong to the selected classifiers against which the GOWA-variant of the similarity classifier is tested) is left outside the scope of this research.

The usability of the GOWA-variant is studied with a credit card client data set found from the UCI Machine Learning Repository, which is a widely used repository of data sets used in Machine Learning related research. The benchmarking will also be done with

two of the same data sets that were used by Kurama et al. (2017) in their research paper. The main empirical results will be the result of using these data sets and thus the results obtained are specific to the tested sets – the results cannot be generalized and the nature of this investigation is exploratory.

In accordance with the focus of this work, the GOWA-variant of the similarity classifier will be introduced and discussed in more detail than the other classifiers used in this thesis. The focus is quite strict, but necessary, as otherwise the work would branch out enormously.

1.3 Research questions

The main research question for this thesis is, "whether the GOWA-variant of the similarity classifier is a useful classifier to be used in the business context".

To study this question this thesis will answer the following sub-questions:

1. What previous academic literature exists on similarity-based classifiers and what are the results found in the said previous research?
2. How do similarity-based classifiers work and especially how does the GOWA-variant of the similarity classifier work?
3. How well does the studied similarity-based classifier(s) function in comparison with other selected classifiers?

In addition to these three sub-problems, this research will construct simple measures to rank classifiers, as there are multiple criteria on which the performance of the compared classifiers can be measured.

1.4 Structure of this thesis

This first chapter introduces the business analytic field and gives motivation for this study. Research questions and scope for this study were also presented in the previous sub chapters.

The next main chapter Methodology will explain the research philosophical groundings for this study. The research paradigm which mostly influences this research work is a design science paradigm and more specifically the focus in this study is in the evaluation part of the model creation. In addition to philosophical discussion, also the research strategy and selected research approaches are presented in that chapter.

In order to be able to do the benchmarking of classifiers, which is the subject of this study, some general understanding of classifiers and machine learning process needs to be gained first. This background knowledge is needed in order to be able to execute the benchmarking work and to analyze the results. The knowledge is first gained by introducing the main terminologies for this study and explaining how there relate to this study. This is done in chapter 3 Theoretical background. In this same chapter also the main subject of the study, similarity classifiers and more specifically the GOWA-variant of the similarity classifier is introduced. Literature review will aim to answer one of sub research question about the previous research work done about similarity classifiers in business context. The literature review will produce a general understanding about similarity classifiers in business analytics and hopefully point out some research gaps which this thesis aims to fill.

The benchmarking of classifiers is done as a numerical testing study in chapter 4. The study is performed with MATLAB software. There are three different data sets that are used for classifier learning and testing process. Two of the data sets are same that Kurama et al. (2017) used in their paper to gain better comparability. The third data set is a credit default data set, which relates better to business context. It is also more complicated than the other two, and could probably gain more insight to comparison of the GOWA-variant of the similarity classifier against the reference classifiers.

In MATLAB there is a set of classifier algorithm implementations available in the "Statistics and Machine Learning Toolbox" that accompanies the software. The original code for the GOWA-variant of the similarity classifier has been available for conducting this research from the original authors. There has been no need to create the code for the algorithms tested in this research from the scratch - this is typical for this kind of work. There is a lot of work connected to finding the optimal parameters to be used for each one of the algorithms. Also the data that is used needs to be pre-processed so that it can be used with the algorithms. Also the separation between training and test sets of data needs to be done. The used evaluation-criteria needs to be determined and the actual goodness measures need to be implemented. The tested algorithms need to be ranked, and the results visualized. The chapter where the benchmarking of the classifiers is done contains a description of how the actual work with MATLAB is performed and the results of the numerical tests are also presented. The classifiers that are used for benchmarking are also shortly presented there.

The conclusion section will summarize the results of the case study, go through the answers to the set research questions, and suggest areas for possible future research.

2 METHODOLOGY

2.1 Philosophical Position of the Research

Research philosophy is a term that refers to a collection of beliefs and theorems about the way knowledge is developed and adopted by a professional community. This essential set of conceptions allows associates of a research community to share similar insights and to participate in common collective practices called a paradigm. A research paradigm serves as a guide for a research and normally contains assumptions about the human knowledge (epistemological assumptions) and how it is acquired, assumptions about realities and the physical word you meet in your research (ontological assumptions) and the scale and the ways your own values influence your research (axiological assumptions). (Saunders, Lewis, & Thornhill, 2009).

Ontology is a theory that describes our view of the nature of reality; i.e., the assumptions about how the word is made up. One aspect to ontology is a division between objectivism and subjectivism, of which objectivism is normally related to quantitative research and subjectivism to qualitative research. Objectivism as an ontological view point proposes that the existence of the world is independent of people and their actions. (Eriksson & Kovalainen, 2015). In this thesis the ontology of the research is objective; the researcher has no influence on the data that is analyzed. Also the used machine learning algorithms that will be used in the analyzing process are not affected by the user, that is, in this case the researcher. The research is completely repeatable by another researcher, the outcome of the research does not include a subjective view of the researcher and the outcome will be quantitative measurable and therefore objectively evaluable.

Epistemology is closely related to ontology and it explains how knowledge can be discovered and argued for. Epistemology defines the suitable ways of enquiring knowledge and defines what knowledge is. It also proposes the sources and limits for scientific knowledge and therefore offers the basis for scientific practices and processes. (Eriksson & Kovalainen, 2015). Epistemology mirrors the relationship between someone as an investigator and the object of the investigation. In this thesis knowledge is developed through making. Knowledge is constructed based on the context, i.e., it depends on the data set. The artifacts themselves are not actually created in this thesis, but the evaluations of the artifacts and benchmarking them against each other with different data sets will gain knowledge of their usefulness in a given context. The resource of this study is the data which is measurable and quantitative; it is considered real and not affected by anybody's opinions or feelings. Although it could be argued that the generation of the data set could be biased, somebody has selected what features are selected and how the

data is represented. But in this thesis the used data is considered to be real data, and the outcome of the research, the performance figures of the classifiers, is a single outcome of the truth. Therefore the philosophy behind this research can be consider as a positivism research philosophy.

Axiology is a theory of values, i.e., what values does the researcher have related to the subject of the study (Hevner et al., 2004). In this study, a pursuit of a fair comparison of the classifiers can be seen as one of the research values. Understanding the basis of the selected classifiers and their usefulness in business related problem solving can also be seen as a value for this research.

The above mentioned philosophical groundings set the basis for the research perspective or paradigm for this thesis. In one way this study can be seen as a positivist research. The world (selected data sets) is observed in a neutral and objective way. The aim is to discover some general relationships in the data sets by using the classification algorithms. The tests performed and thus the observations made in this thesis are fully repeatable and are not affected by the personal view of the researcher. This thesis also relates closely to the design science research paradigm, presented by Takeda et al. (1990) and further developed by Vaishnavi & Kuechler (2004). This paradigm is a problem solving paradigm which concerns research of man-made constructs or artifacts, their generation and usage, implementation and evaluation. The Design Science Research Process Model is depicted in Figure 1. In that model the design science research effort is divided into five development steps: Awareness of Problem, Suggestion, Development, Evaluation and Conclusion. This thesis will roughly follow this process, with the main focus being in Evaluation and Conclusion steps. The awareness of the problem comes with the selected data sets. The practical part of the thesis will contain also the development part, but the artifacts that are used there are taken from prior work outside of this thesis.

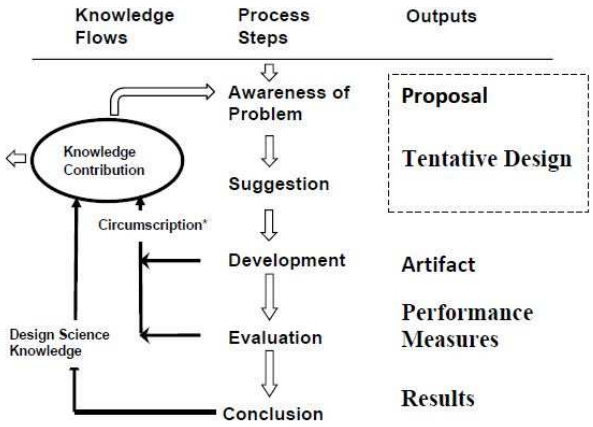

Figure 1 Design Science Research Process Model (Kuechler & Vaishnavi, 2004).

Hevner et al (2004) also defines design science as a problem solving paradigm. They state in their article that knowledge and apprehension of a problem domain and its solutions are achieved by building a designed artifact and applying it to a real problem domain. The aim of this study is not to build the artifacts, but to apply them into real data sets and to evaluate their performance, which is a part of the design science research cycle.

So basically in this research an information technology innovation is taken into use (the GOWA-variant of the similarity classifier) and its usefulness is evaluated in a business problem solving environment and then it is benchmarked against other classifiers. Wang & Wang (2010) present quite nicely how design science research in information systems relates to information technology innovations. This relation is depicted in the Figure 2.

Figure 2. Alignment of Design Research and IT Innovation (S. Wang & Wang, 2010).

This thesis is positioned into the business problem solving box in the Figure 2, evaluating artifacts. The evaluation of artifact also relates to the Knowledge Contribution area, since the aim of this study is also to investigate the usefulness of the similarity classifiers and add knowledge of their suitability to solve business problems.

2.2 Research Approach

Research approach is quite commonly divided into qualitative and quantitative research. Quantitative research is defined as a research that focuses on numbers or any data that is in numerical form. By describing and interpreting research objects with numbers mathematically or statistically, the researcher analyses the data and aims to a result that is unbiased and can be generalized to a larger extent and repeated by another researcher. In this thesis the data which is analyzed is numerical and it will be analyzed with computational methods by using classification algorithms. The results from the benchmarking process are also numerical and are easily comparable to each other. The

performance figures of similarity classifier compared to other classifiers are numerical and the opinion of the researcher has no effect to the results. Therefore this research can be categorized as quantitative research. On the other hand the evaluation of the classifiers can also be seen to be partly qualitative research, since it is descriptive by nature and the selection of the used classifiers are done by the researcher, and this has obviously some effect for the evaluation process. So although this thesis is mainly quantitative, it has some features of qualitative research as well.

In design science research, when an artifact is generated, one way to evaluate whether the artifact constitutes research, is to compare it with existing artifacts that are used for the same, or similar, purpose. Benchmarking is one way to discover the best performance from among the compared artifacts. The measurement of performance is done by using a specific indicator, in this case the classification accuracy, among others, which results a quantitative metric that can be used in comparison with other classifiers. In this thesis the GOWA-variant of the similarity classifier is benchmarked against to a set of generally used classical classifiers. The benchmarking process is performed in a controlled environment; the same data sets are used for each classifier.

In addition to quantitative and qualitative, research approach can also be classified as inductive or deductive. Inductive reasoning means that theory is developed based on the data that has been collected and analyzed, i.e., it aims to formulate general concepts from specific observations. In deductive reasoning a hypothesis and a theory is first developed and then tested in the research, i.e., the aim is to predict what the observations should be based on a theory. Deductive reasoning goes to opposite direction as inductive reasoning; it goes from general, a theory, to details, that is, to the observations. According to Saunders et al. (2009) deductive research highlights following features: moving from theory to data, an assortment of quantitative data, a highly organized approach and researcher objectivity of the researched subject. Although this thesis is not purely a deductive research, the above mentioned points about deductive research approach can be related to this study thus making this study more deductive than inductive. This is depicted in Figure 3, where a general illustration of a classification model building process is expressed.

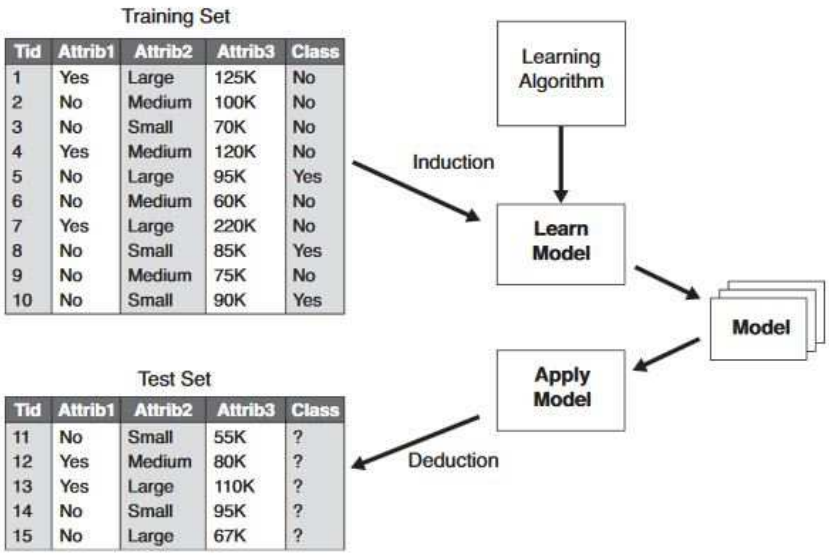

Figure 3 General illustration of building a classification model. (Tan, Steinbach, & Kumar, 2006).

From the figure it can be seen that the learning phase of the machine learning process where the classification algorithm uses a training set of data for learning is an inductive process, whereas the testing part of the model is seen as deductive process. This thesis does both parts of the process, but the main interest is in the model application. Therefore it is justified to note that this research is a deductive research.

2.3 Research Design Overview

Research Design is a general plan or research strategy, which will explain how the research work will be executed and how the research questions will be answered. Objectives for the research should be derived from the research questions and the sources for the data that is used in the study should also be defined. The aim for a research design is to show that it has been thought out well why this research is done and how it answers to the research questions in an effective manner. It is also important to take into account the amount of time that is available for the study and the existing knowledge, since these will have an effect for the research work. (Saunders et al., 2009). Research design can also describe a framework that is the basis for the research work and what will be used as a guideline for answering the research questions. This section will concentrate on defining a framework that has been used as a background for this study.

Mitroff et al. (1974) present a problem solving model, which can be used as a framework for solving business analytic problem, such as a credit default detection problem, which is analyzed in this thesis. In Mitroff's model the problem solving process is divided into four steps as illustrated in Figure 4.

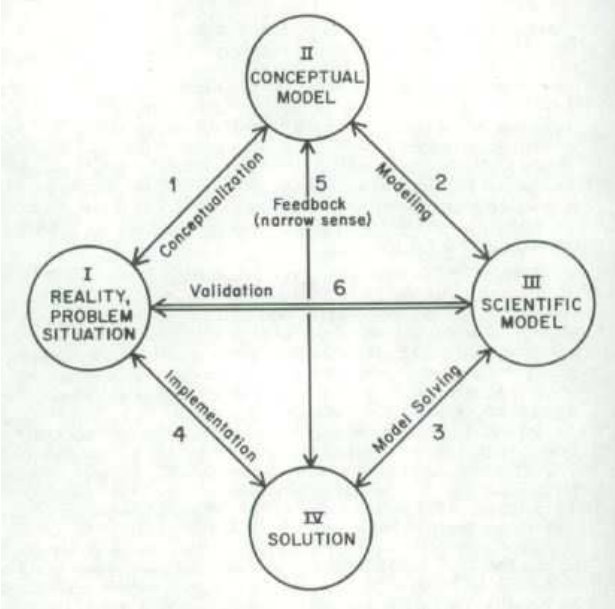

Figure 4 A systems view of problem solving (Mitroff et al., 1974).

The first step is defining a business problem, for example distinguishing fraudulent transactions from the legitimate ones. Steps two and three are model building phases. This thesis concentrates on the arrow six in the model, the validation part of the selected models; which are in this study the different classification algorithms.

Purpose of this study is to analyze how well the GOWA-variant of the similarity classifier performs against a set of classical classifiers. The analysis is done with existing data, which can be found from a general database for machine learning problems; The UCI Machine Learning Repository (Lichman, 2013). This database is widely used by machine learning researchers and practitioners. It is hosted and maintained by the University of California and it is freely accessible. The other purpose for this research is to study how well similarity-based classifiers suit for business analytic problems. Therefore the main dataset that is used for the classifier benchmarking is a business related data, a credit default detection dataset. Most of the earlier studies of similarity-based classifiers are done with medical data. The benchmarking is done with MATLAB, which is a numerical computing environment. MATLAB was chosen as a tool to perform the modeling of classifiers and the benchmarking, since there are numerous algorithms already implemented in MATLAB with rich documentation. The research work can therefore focus on modeling itself, not on the programming. MATLAB is easier to take into use as a first machine learning related tool compared to a more programming oriented approach such as R or scikit-learn for the Python. Both of those other two methods are also widely used by machine learning practitioners, R mainly by people who have a statistical background and Python by people who have background in computer science. MATLAB is suitable for a person with business school background since it does not

require that much programming experience. There are also good visualization properties in MATLAB which can be used for presenting the results.

For a theoretical background the similarity-based classifiers and mainly the GOWA-variant of the similarity classifiers are introduced in this thesis. The literature review is done to study how widely similarity-based classifiers have been used in solving business problems and to find for what kind of problems they could be used in the future.

The practical research part of this thesis is based on modeling and evaluation. The evaluation is done numerically, so results are not affected by the researcher, instead they are fully repeatable by another researcher. The results should show how well the GOWA-variant of the similarity classifier works in relation to other selected classifiers. The other classifiers are selected based on both the literature; the intention is to use classical classifiers that have been frequently used in other application, and the availability of the models in MATLAB.

3 THEORETICAL BACKGROUND

3.1 Terminology

This section defines some of the key terminologies related to this thesis and explains why these are important for this study and also in modern business environment.

Machine Learning

According to Samuel (1969) machine learning is a field of study that gives computers the ability to learn without being explicitly programmed. Machine learning is a field of science related to exploration and generation of algorithms that can learn from data and make predictions on it. Machine learning is a part of artificial intelligence and is focused on computer programs that have an ability to learn from experience by analyzing data, making observations and interacting with the world and adapting accordingly. The learning process will happen automatically, without any human intervention. (Varone, Mayer, & Melegari, 2018). That is what separates machine learning from traditional purely static and explicit programming.

Machine learning will use a collection of methods to extract a predictive model from accessible data. Thus the aim is to predict future events; which is a valuable asset in many business related problems and what the traditional static algorithms cannot do. Machine learning algorithms are constantly improving over time as more data is captured and merged to the system. The new data can be used to test the predictions that have been made based on the previous data and as a result the algorithm can adjust to the new information and become more accurate in time. Thus the machine learning algorithm should evolve over the time. (Provost & Fawcett, 2013).

In general machine learning algorithms are able to look for patterns in data that are not possible to be detected by humans and doing that in scale. Good application areas for machine learning are computing tasks, where developing an explicit algorithm with decent performance is difficult, or even impossible. Such areas are for example credit scoring, credit default (which will be investigated further in this thesis), email filtering, effective web search, price forecasting, bond classification, speech recognition, understanding a human genome, detection of malicious transaction, online recommendation offers, computer vision, or self-driving cars. (Rouse, 2018), (Marr, 2016). The list is quite widespread; in fact machine learning is so ubiquitous today that people probably use some machine learning algorithm several times in a day without even noticing it. Machine learning has become popular, because of the growing volumes and variety of features of available data, cheaper and more powerful computational resources, and more affordable and powerful data storing capabilities.

All the above mentioned facts mean that organizations are able to develop models effectively and use those models to analyze bigger amounts of more complex data than ever before, and deliver more accurate results faster and in a very large scale. With these models organizations are able to identify profitable business opportunities easier than before and gain a competitive edge and also avoid risks better.

Supervised learning

Machine learning algorithms can be divided into three groups, supervised, unsupervised, and reinforcement learning. Unsupervised learning interprets unlabeled data only based on input data. It tries to find some structure within the data and to figure out what kind of relationships there are in the data and what kind of clusters, or segments, can be created. Whereas with supervised learning the goal is to learn from the past to be able to predict the future, based on both the input and output data. The labeled examples from the past data are used for creating a machine learning algorithm which is then used for labeling the new data. The labels could be for instance simple yes/no in email filtering algorithm, meaning that email is either spam, or not spam. In other words, if a specific target can be provided for the data mining problem, it is a supervised problem. But there has to be enough data available with the target values provided along with the data before a supervised learning algorithm can be applied. (Provost & Fawcett, 2013). Reinforcement learning is the third option, and it falls somewhere in between supervised and unsupervised learning.

Supervised learning problems are usually divided into regression and classification problems. In a regression problem, the output will be a continuous numerical value. A regression problem could an investigation on how much a customer will use a certain service. Whereas in a classification problem there is a discrete output, or categories to which input variables are mapped into, like whether a customer will respond to a marketing campaign, or not. (Provost & Fawcett, 2013). First, in a learning phase an algorithm receives a set of input instances together with corresponding correct output values (class labels) and it uses them to find patterns and relationship between input and output values. The algorithm then compares the actual values to the correct output values, and adjusts and modifies the model accordingly when it finds errors. Then when a new unlabeled data is fed to the system, the classifier (=machine learning algorithm created in the learning phase) can predict the output values by using the patterns that were detected in the learning phase. A supervised machine learning process is illustrated in Figure 5.

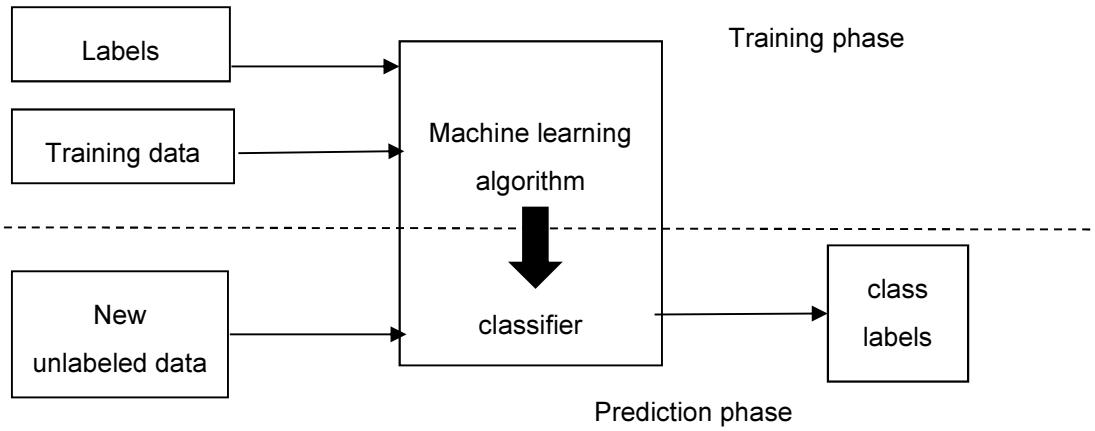

Figure 5 Supervised Machine Learning process.

Supervised learning is suitable for application where historical can be used for predicting future events, for example detecting fraudulent credit card transactions.

Classification

Allaby (2013) defines classification as "Any scheme for structuring data that is used to group individuals". The "individual" can refer to an entity of which there is data available such as a consumer or a customer or a non-living entity like business (Provost & Fawcett, 2013). In machine learning these entities are often called as "instances". Classification can also be defined as one example of a pattern recognition problem, where there is a need to map some output value to a given input value.

Classification is a supervised data mining task, which aims to predict to which of a known set of categories, or classes, each instance in a population belongs to. Usually the classes are mutually exclusive. (Provost & Fawcett, 2013). This classification is done by building a model based on one or more numerical and/or categorical explanatory variables or feature or attribute. The terminology varies, but feature is probably the most used term related to classification problems in machine learning.

The classification algorithm, called "a classifier", is first trained with a training set of data, which includes the correct class labels for each observation. After the training the classifier can be used for making predictions to which category a new, previously unanalyzed data belongs to. A classification algorithm can give an answer to a question like: Is this credit card transaction fraudulent or not? Is the found tumor malignant or benign? Will this loan application default? What is the correct category for this article? What movie categories this online customer likes? So in other words, the aim is to map an input in a specific input space to a defined classifier output space.

There are several classification algorithms and methods for predicting the class labels. But all classifiers have a common view that instances, which share a common region in a target feature-space should be similar. What differs between the various classification methods is how these regions are characterized and discovered.

Each instance can be represented with a feature vector; features can be for instance attributes that characterize different credit card applicants: age, gender, demographic, monthly income, marital status, education, amount of credit applied etc. Feature vectors define a feature space and the closer the objects are in that space, the more similar they are. The problem is how to measure how similar two objects, like customers or companies, are, and what does it actually mean that they are similar. But once the similarity can be defined and measured, this information can be used in decision making and appropriate actions can be done based on the classification definitions. For instance online ads can be targeted better or companies can use the information to target sales force to corporate customers that are similar to their known good beneficial customers.

Kotsiantis (2007) presents a machine learning process applied to real world problem that is depicted in Figure 6. The figure shows that the process is an iterative process, which could require many rounds, before a good classifier suitable for the given problem is created. Before any classification can be performed there needs to be enough data with required explanatory variables available. In most cases some pre-processing is performed before data can be fed to the machine learning algorithm, a classifier in this case. Evaluation is a vital part of the learning process. Only a part of the available data with labels are used as training set, the other part is used for evaluation of the classifier. When the evaluation shows that the classification rate is at acceptable level, the classifier can be used for predicting class labels for a new unlabeled data.

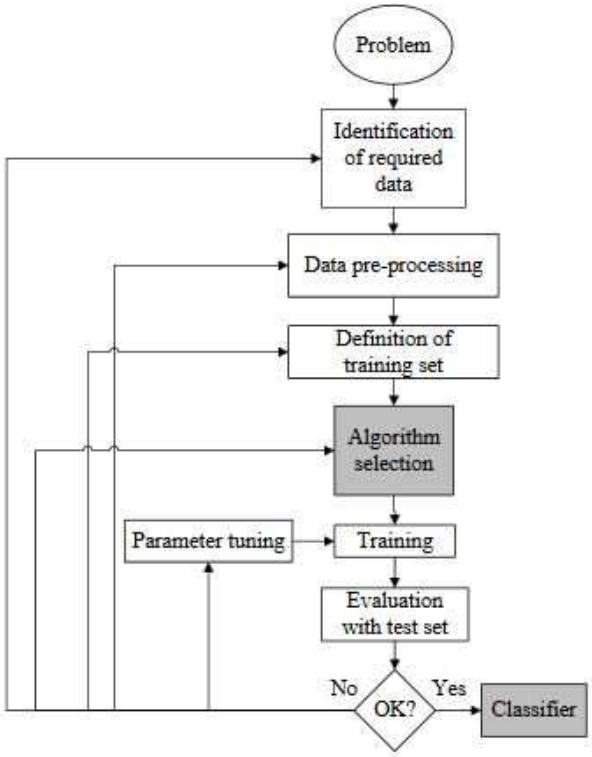

Figure 6 Machine Learning process (Kotsiantis, 2007).

The selection of an algorithm is not a trivial task, but it is a critical step in the whole process, since it greatly influences the outcome. There are a wide variety of different classification algorithms available, which are based on different techniques. For instance classification can be based on similarity or distance measures (e.g., k-nearest neighbor), probability assumptions (Bayesian networks), frequency of observations (decision trees), or definition of boundaries (neural networks) etc. Many classifiers use some distance indicator between observed instances to define the classes. Usually it requires many trials with different algorithms and with different parameters before the most suitable algorithm for a given problem is found. The quality of a classifier is generally evaluated by the classification accuracy, i.e., the correct classifications versus all predictions.

3.2 Similarity-based classification

According to Janusz (2008), researchers have been interested in the notion of similarity for many years. If we are able to separate similar objects from those which are dissimilar while deciding the class label for the object, we are able to produce an accurate classification and we are able to detect unusual behavior or situations. While the human mind is capable of learning similarity relations from examples, it is not that straightforward task to be performed with computers and algorithms. That is why there is no single method for building similarity models, which would be suitable for all applications of complex problems in various areas.

A lot of different methods have been used for creating models which would be able to define the relation between similar and dissimilar objects and to build a model with good predictive power. In many of the models the items are treated as points in a metric space of their features and the similarity between the objects is a decreasing function of the distance between them, i.e., two items are considered similar if they are close enough to each other in that feature space. It is natural to think that objects that are likely to be similar to each other have many similar attribute-values. (Janusz, 2008).

The assumption about closeness between similar samples is a reason why standard metric-space classifiers are the most common classifiers. They use numerical features to define the training and test instances. The features are represented as d-dimensional vectors in a Euclidean space. Metric-space classifiers generally assume that the pair-wise similarity between samples, or instances, is represented by a metric distance function, such as the Euclidean, Manhattan, or Chebyshev distance, which measure the physical distance between objects. Distance measures are used by the classification algorithm to place the similar samples into the same class, while distant sample points are placed into different classes (Shirkhorshidi, Aghabozorgi, & Wah, 2015).

Two objects can be similar to each other even though they do not necessary bear common distance measure features, such as symmetry and subadditivity. This is especially observed in complex situations with multi-variant samples. The reason for this is that complex objects are similar in some ways (attribute-values) and dissimilar in others. The reliance between local and global similarities is nonlinear and thus harder to model. This dependency needs to be learned from the data. (Janusz, 2008).

If data samples are multi-variant and contain both numeric and non-numeric data, it is not easy to represent the data in the Euclidean space. In these cases the distance functions are not able to completely represent the relationship and the similarities or dissimilarities between the instances. All data cannot be represented in a geometric feature space with continuously-valued numbers and therefore the classifiers that rely on metric similarity of distance-functions are not applicable either. It could also be in some applications that the principal features of the data are not accessible, instead only the pair-wise similarities could be observed. (Cazzanti, 2007).

Similarity is a more general term than distance and may therefore be more suitable for cases with complex multivariate data, or for cases where all data is not numerical. A similarity measure is a numerical measure of how much two data instances resemble each other. Similarity measures are often defined between values 0 and 1, where value 0 means that there is no similarity and value 1 means that two instances are completely similar. So instead of using distances on the basis of the classification, similarity-based classifiers use an alternative learning method, which uses similarities between test and training samples for estimating class labels. (Cazzanti, Gupta, & Srivastava, 2009).

It is the same thing with similarities as it is with distances; there are several ways to measure the similarity. Also with multivariate data there are various ways how the similarities of different features are aggregated to form the finishing similarity for a particular instance. The choices for a similarity measure and aggregation function have naturally an effect on the accuracy of the classifier. The aim for this study is not to compare different similarity measures or aggregation functions. The similarity classifier and the aggregation method that are used in this thesis are the same as Kurama et al. (2017) use in their research paper. The purpose of this thesis is to compare this particular similarity classifier to a set of other classifiers and to find out how well it performs compared to other classifiers.

In order to be able to understand the GOWA-variant of the similarity classifier introduced in the next chapter one general example of non-metric similarity functions that can be used for pattern recognition and classification is introduced next. The example case is Tversky's linear contrast model (Tversky, 1977). This model assumes that each sample can be represented as a set of features. This model is a binary model, each sample either has a selected feature or not. The similarity function is an increasing function of overlapping feature set and decreasing functions of feature set of differences (Chen et al.,

2009). An example of Tversky's model is presented in Figure 7. In this example each sample is presented with four facial features: eyes, nose, month, and hair. One sample for both of the classes is defined as the center point of that class. Class geometric center, the centroid, is typically defined as a sample which has the maximum similarity with all the other samples in that particular class. The similarity between two samples is calculated to be a number of facial features that these two samples have in common. A sample belongs to the class with the most similar centroid sample. (Cazzanti, 2007).

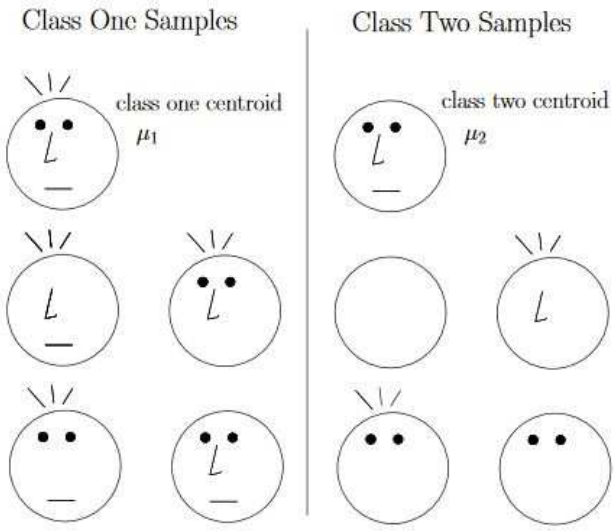

Figure 7. Similarity based classification example. (Cazzanti et al., 2009).

Tversky's similarity model has been successful in explaining human similarity judgments in several similarity-assessment problems, but it has also been useful in other fields than psychology (Cazzanti, 2007).

There are also many other similarity functions defined to measure similarity; for example a value difference metric is designed to measure the distance between samples that are defined with non-numeric features or Lin's similarity measure that is based on information content of feature vectors to name a few (Cazzanti, 2007). The next chapter defines the similarity functions and classification method used in this thesis.

3.3 GOWA-variant of the similarity classifier

The main idea of the similarity classifier is to first create "ideal vectors", which are representatives of each of the target classes. These ideal vectors will then be used for calculating similarity between a new item and those representative vectors and to select correct class label for that new item. (Kurama, Luukka, & Collan, 2016a). Similarity-based classifiers contain an aggregation step, in which the similarities of single features of multi-

dimensional data items are combined. The GOWA-variant of the similarity classifier is a similarity classifier that uses a generalized ordered weighted averaging (GOWA) operator in aggregation (Kurama et al., 2017). The overall goal for a classifier is to place new samples into a set of selected classes as accurately as possible. First, a set of representative vectors, or centroid samples, needs to be selected for each class. Each representative vector includes a set of measured features, which are usually first scaled to a unit interval [0, 1] to ease the calculations. Then a new sample is compared against these representative vectors and it is labeled with the class label of the representative vector with which the new sample has the highest similarity value (to which the new sample is most similar to). Similarity is calculated in a way that the similarity of a sample with itself is equal to 1. This means that a similarity value against other samples should be less than or equal to 1.

The selection of which aggregation operator is used in a similarity classifier is an essential decision, since it directly affects the classification accuracy of the final classifier (Kurama, Luukka, & Collan, 2016b). Aggregation operators are mathematical entities, the purpose of which is to reduce the set of numbers to a unique representative and meaningful number (Detyniecki, 2001). That is, aggregation operators are used for combining N numerical values into a single numerical data value. The most common aggregation operators are the arithmetic mean and the weighted mean. The weighted mean differs from the arithmetic mean by allowing a weighting of different data items according to their significance. (Torra & Narukawa, 2007). In a classification problem there are several predictors (or parameters) that are used for determining to which class this particular item belongs to. In similarity classifier the similarities of these decision criteria values (i.e., the parameters) are combined to form a unique similarity value which can then be used to determine the target class. An aggregation operator is needed for that operation. In the following definitions of a few aggregation operators are introduced. A simple and most common aggregation operator, the arithmetic mean is expressed mathematically as:

$$\boldsymbol{f_{aM}(a_1, a_2, \dots, a_n) = \frac{\sum_{i=1}^{n} a_i}{n}} \qquad \boldsymbol{(1)}$$

Weighted mean can be expressed mathematically as:

$$\boldsymbol{f_{wM}(a_1, a_2, \dots, a_n) = \sum_{i=1}^{n} \omega_i a_i \ \ , where\ \omega_i \in [0, 1]\ and \sum_{i=1}^{n} \omega_i = 1} \quad \boldsymbol{(2)}$$

Where ω_i equals to the weight, or the relevance, of the a_i information item. Weights should all be positive and their sum should be one. The weight is typically understood as a defined significance factor for each element.

Ordered weighted averaging operators were introduced by Yager (1988) and they can be expressed mathematically as:

$$f_{owa}(a_1, a_2, \dots, a_n) = \sum_{i=1}^{n} \omega_i \, a_{\sigma(i)}, \quad \omega_i \in [0,1] \; and \; \sum_{i=1}^{n} \omega_i = 1 \quad (3)$$

where σ is a permutation, which orders the elements in descending order: $a_{\sigma(1)} \leq a_{\sigma(2)} \leq \cdots \leq a_{\sigma(n)}$. This reordering step is an essential feature for this aggregation operator. In OWA aggregation the weights are not connected with a specific argument, but with the ordered positions of the arguments (Yager, 1996). Different OWA operators are distinguished by their weighting function (Fullér & Majlender, 2003). OWA operators provide a parametric family of mean type aggregation operators, such as the arithmetic average, the median, the minimum, and the maximum. In order to attain these specific operators, the weighting vector needs to be defined accordingly, for instance the minimum operator is selected by setting $\omega_1 = 1$ and other weights to zero. (Detyniecki, 2001).

There are number of different approaches that have been developed for determining the related weights for the OWA operator. Zhou & Chen (2014) have collected a few alternatives methods for a weight generation in their paper. For instance a learning technique that is built on observed data, exponential smoothing process that produces exponential OWA operator and weights, maximum entropy model that conveys the OWA operator weight problem as a constrained nonlinear optimization model, and a quantifier guided aggregation that computes weights using linguistic quantifiers.

The aggregation operator used in the similarity classifier used in this thesis, a generalized OWA operator, is an extension to the OWA operator, also introduced by Yager (2004). The GOWA operator adds an extra parameter to the OWA operator which controls the power to which each argument value is raised. GOWA operator is defined as:

$$GOWA(a_1, a_2, \dots, a_n) = \left(\sum_{j=1}^{n} \omega_j \, b_j^{\lambda} \right)^{1/\lambda} \quad (4)$$

where ω_i are the weights (values between zero to one and they add to one), λ is a parameter ($\lambda \in [-\infty, \infty]$) and b_j is the jth largest value of a_i. The choice of a parameter value λ has an effect on classification accuracy and therefore it should be investigated which λ value gives the best accuracy. I.e., the GOWA operator behavior changes as the λ-value changes (Kurama et al., 2017). Another thing that affects the classification accuracy is the choice of how the weights are defined. In the model used in this thesis weights are determined by using quantifier guided aggregation.

Yager (1996) proposed the use of fuzzy linguistics quantifiers for obtaining the weights for the OWA operator. One class of the proposed quantifiers is Regular Increasing Monotone (RIM) quantifiers, which can be expressed using verbally concepts,

such as all, most, many, and there exist. Weights with RIM quantifiers can be determined as:

$$\omega_i = Q\left(\frac{i}{n}\right) - Q\left(\frac{i-1}{n}\right), i = 1,2,\dots,n \text{ and } Q \text{ is the quantifier} \quad (5)$$

Further details about quantifier generation can be found from literature (Yager, 1996) & (Fullér & Majlender, 2003) & (Liu & Han, 2008), but are left outside the scope of this thesis. Kurama et al. (2017) used four different linguistic quantifiers which will produce four different weight generation schemes. The same quantifiers are used for this thesis. The used quantifiers are basic RIM, polynomial, exponential and trigonometric quantifiers and their equations are presented next.

$$Q_1(r) = r^{\alpha}, \alpha \geq 0 \quad (basic\ RIM\ quantifier) \quad (6)$$

$$Q_2(r) = 1 - (1-r)^{\alpha}, \alpha \geq 0 \quad (polynimial\ quantifier) \quad (7)$$

$$Q_3(r) = \left(\frac{e^r - 1}{e - 1}\right) \quad (exponential\ quantifier) \quad (8)$$

$$Q_4(r) = arcsin(r\alpha) \quad (trigonometric\ quantifier) \quad (9)$$

The equivalence e between two items x_1 and x_2 can be defined as:

$$e(x_1, x_2) = \left[1 - \left|x_1^p - x_2^p\right|\right]^{\frac{1}{p}} \quad (10)$$

If equivalence between two samples are denoted as $e(x_1, x_2) = (e_1, e_2, \dots, e_n)$, where e_i is defined as in equation (10), we get the similarity measure with GOWA operator as:

$$S_{GOWA}(e_1, e_2, \dots, e_n) = \left[\sum_{i=1}^{n} \omega_i b_i^{\lambda}\right]^{\frac{1}{\lambda}} \quad (11)$$

where ω_i is an n-dimensional weighting vector, λ is a parameter for the GOWA operator and b_i is the i^{th} largest value of $(e_1, e_2, \dots, e_n)$. Each of the equations above are taken from Kurama et al. (2017) publication. More details about how the similarity measure is defined can be found in ibid.

3.4 Literature review

Fink (2005) defines literature review as "a systematic, explicit and reproducible method for identifying, evaluating and synthesizing the existing work produced by researchers,

scholars and practitioners". The aim for a literature review is to provide a relevant background of what has already been done in the field of the study and therefore provide a context for the research work. In this chapter the pre-existing knowledge about similarity classifiers especially in the business context is gathered together and a literature review of the selected relevant articles is conducted.

The literature review in this thesis is performed with a structured three-step process presented by Webster and Watson (2002). Webster and Watson claim that an effective literature review forms a sound basis for developing theories and progressing knowledge. In their article, they present an intangible and structured approach of doing literature reviews in Information Systems. The first step includes defining the key-words for the search for the relevant literature. The databases where the search is performed also need to be chosen. Webster and Watson (2002) state in their article, that the leading journals include most probably the major contributions for the field of study. But they also argue that since Information Systems is an interdisciplinary field, the search should not be performed only within the leading journals of IS alone. In this thesis the relevant related disciplines are computer science, operational research and mathematics, since similarity classifiers are especially aimed for fuzzy logic applications.

"Similarity classifier" and "similarity-based classification" were selected as quite obvious search strings for the research. Webster & Watson (2002) suggest that a multidisciplinary database should be used for the search process. Web of Science, a search and indexing service, was selected for this research, since it seeks information from a number of databases related to different disciplines. This reference database was selected to get a wider view and to include other fields than IS to this research, since the field of study is closely related to other disciplines and a search from IS or business related database alone would not have resulted wide enough view on the topic or not enough articles for this review. In fact, the search on Business and Economics related database (ABI/INFORM Collection (ProQuest), EBSCO – Business Source Complete, EconLit (ProQuest) and Emerald) only resulted one article, which shows this literature review should be performed in a multidisciplinary context and also that there exist a research gap in similarity classifiers related to business.

The search in Web of Science reference database resulted in 105 hits, of which approximately half were journal articles and half proceedings papers. Web of Science categorize these papers by research area. That categorization shows that this subject is rather technical, since most of the papers are classified into computer science and engineering research areas. Only 10% of the papers are categorized under a research area, which could be seen as really business-related research. But this categorization gives a too simplified picture about the research in this area. The research in many of the papers is concentrated in the model development, i.e., enhanced classification accuracy and the

concept of the similarity measurement, and not on the application area, where the developed model could be used.

The resulting 105 papers were scanned and irrelevant ones were omitted, which resulted in 76 papers that are to some extent relevant to this research. It was observed that the research in this area is quite focused on a few countries, 56% of the papers originate from Finland, USA or China, Finland being the origin of most published papers (22% of the total 76) in this area. The abstracts and some key points of the 76 papers were investigated to find out what is the main focus and the application area in those papers. The main focus in at least half of the papers is in the concept of similarity and the development of the similarity based classification algorithms. That is, in most of the papers the main focus is not on the application area where the classifier is used, instead they are more focused on the classifier itself and in the improved classification accuracy. The academic interest is focused on modeling and developing more accurate similarity classifiers and finding grounds for using similarity, instead of some other measure, on the basis for the classification. Medical diagnostics, image (2D and 3D), voice and text recognition are widely used application areas for similarity classifiers. These could be naturally applied to business cases in relevant area, but the articles themselves do not directly demonstrate how decision-making in business can benefit from the similarity classifiers that are introduced in the papers.

One quite obvious business-related application area for classifiers in general is credit scoring. Within the papers that were scanned in this review, there were three papers that analyze credit card data. One of the reasons why there is a scarce amount of research on similarity classifiers related to business analytics problems is probably the lack of appropriate data. Quite many of the 76 papers use data from the UCI machine learning data repository (Lichman, 2013), which is an open source data repository widely used among machine learning researcher and practitioners. That repository does not have too many business-related data sets available. This indicates that there is definitely a room for more research for similarity classifier usage in business problems.

Based on the resulting papers, it seems that the most utilized application area for similarity-based classification is medical data and medical diagnostics since nearly half of the papers used medical data as a case study. There were just a couple of papers that can be directly related to solving business problems. Of course it also depends how the business problem space is defined. Better diagnostics will naturally give a competitive edge for a medical organization etc., but we consider business problems more generally belonging to the domain of business. According to the definition from Techopedia (Janssen, 2018) business analytics refers to all methods and techniques that are used by an organization to measure performance. In this thesis business analytics is seen as data-driven decision-making process and it should be used by the organizations to identify weaknesses in existing processes and to enable future growth. Business analytics

generally applies statistical algorithms to historical data and based on that make predictions about future sales, product performance, services, customer segmentation, website usage etc.

Based on the definition above there were only a couple of papers that could be directly related to business and business analytical problem solving space. Luukka et al. (2010a) applies similarity classifier in bankruptcy analysis, Skabar et al. (2013) uses similarity classifier while forecasting financial time series, credit scoring is investigated in two papers (Kurama, Luukka, & Collan, 2015), (Steffens, 2005) and in Imran's et at. (2016) paper similarity classifier is used in malware detection which could be utilized in various business cases. Wang et al. (Z. Wang et al., 2016) investigate social media with emotion sensing, which could be applied to many businesses since the use of social media as an advertisement channel is growing rapidly all the time. Each of these six papers were analyzed in more detail and the main points of these five papers are gathered in Table 1.

Table 1. Literature review papers.

Title	Nonlinear fuzzy robust PCA algorithms and similarity classifier in bankruptcy analysis
Authors	Luukka, Pasi
Purpose	Bankruptcy prediction in one of the most interesting task and research subject in financing (Luukka, 2010b). Bankruptcy prediction and credit scoring are also popular application areas for traditional classifiers. In this article bankruptcy analysis is carried out by first preprocessing the data with two different principal component analysis (PCA) algorithms and then applying the resulted data to similarity classifier to differentiate whether the credit card application should be accepted or not. The focus in this paper is more in principal component analysis than in the similarity classifier itself.
Research methods	The data that was used in this research was downloaded from the UCI machine learning data repository (Lichman, 2013) and the used algorithms were developed with MATLAB software.
Conclusion and results	The best classification accuracy achieved in this research was 88.39% with one of the investigated principal component analysis algorithm and similarity classifier. As a comparison, the accuracy was lower if it was analyzed without PCA and with traditional PCA. Another notable issue was that the best results were achieved with quite low dimensions, good accuracy was achieved with the first three principal components. This means that complex datasets can be significantly simplified which reduces required computational time and resources. (Luukka, 2010b).

Title	Fine-Grained Sentiment Analysis of Social Media with Emotion Sensing
Authors	Wang, Zhaoxia; Chong, Chee Seng; Lan, Landy; Yang, Yinping; Ho, Seng Beng; Tong, Joo Chuan
Purpose	Social media is nowadays used widely and it contains a huge amount of human generated text; opinions, feedbacks and critiques that reflect attitudes and sentiments towards different things. This paper describes an adaptive fuzzy similarity-based analytics engine, which classifies text messages into sentiment categories (positive, negative, neutral and mixed) and is also able to prevail emotion categories (such as satisfaction, happiness, excitement, anger, sadness and anxiety). I.e., the purpose is to offer means to understand the public sentiment. (Z. Wang et al., 2016).
Research methods	There is a social adaptive inference algorithm that simulates human expression and emotions in online social context. A fuzzy similarity rules are used for handling sentiment classification.
Conclusion and results	The research presents an analytics method for handling fine-grained sensing and emotions classification. There are a lot of opportunities where this can be applied. It could benefit healthcare, corporate and public and private sectors to understand their customers better and to improve their products and services.
Title	Direction-of-Change Financial Time Series Forecasting using a Similarity-Based Classification Model
Authors	Skabar, Andrew
Purpose	Financial time series forecasting is a popular topic in finance. Traditionally future values of a time series are predicted based on the past values and the accuracy of the prediction is estimated by comparing the predicted values to the actual realized values. In many cases the ability to predict the direction of the change (up/down) is more important than the magnitude of error in the prediction, which makes this a suitable classification problem. In this paper a similarity-based classification model is introduced which is used for predicting upward/downward movements in stock market. (Skabar, 2013).
Research methods	The model is applied to daily closing prices of the Dow Jones Industrial average over a 20-year out-of-sample period from 1 January 1989 to 3 December 2009. The performance of the similarity-based classifier is compared to a logistic regression and multilayer perceptron (a linear binary classification algorithm).

Conclusion and results	The overall prediction accuracy was only a little over 50%, which is not much better than pure chance. But it is not the whole truth, since the prediction accuracy varies a lot over the input space. But all in all, the researcher did not find any substantial difference in performance between similarity-based and MLP classification approaches with the used dataset. (Skabar, 2013).
Title	Partial and Vague Knowledge for Similarity Measures
Authors	Steffens, Timo
Purpose	The purpose of this paper is to introduce an enhancement to the similarity-based classification by using virtual attributes from imperfect domain theories. Virtual attributes are attributes that are not directly represented in the test data but they can be derived from the already existing attributes. (Steffens, 2005).
Research methods	The method is applied to two datasets, one of them is a Japanese credit card data from UCI Machine Learning database. The focus in this paper is in intermediate attributes, which are good candidates for virtual attributes and can be added to the similarity measure and by doing so possible enhancing the classification accuracy (Steffens, 2005).
Conclusion and results	This article shows that even an imperfect domain knowledge can enhance the similarity-based classification. This means that domain knowledge does not need to be complete and fully accurate in order to be still useful.
Title	Malware classification using dynamic features and Hidden Markov Model
Authors	Imran, Mohammad; Afzal, Muhammad Tanvir; Qadir
Purpose	As digitalization affects most of the businesses today, organizations also need to pay more attention to the information security. The number of new malware threats has increased notably, triggering loses of billions of dollars globally. Therefore malware needs to be identified in order to provide corrective and defensive actions towards it. This paper uses a similarity-based classifier to identify new malware by classifying it to a set of previously defined malware classes, since most of the new malware resembles some of the previously recognized malwares. (Imran et al., 2016).
Research methods	This paper compares two classification methods (similarity-based and maximum likelihood), which both are based on Hidden Markov Model, for classification of malware.

Conclusion and results	Similarity-based classification was found to perform better than maximum likelihood classification scheme. It was discovered that when making the decision about which malware family a new malware belongs, the pattern of the sample's similarity with all malware families should be taken into account. The decision should not be made on the basis of the closest match of with a single malware family. Neither one of the used classifiers were able to detect benign samples effectively, but similarity-based classifier performed slightly better on this tasks also. (Imran et al., 2016).
Title	Credit Analysis Using a Combination of Fuzzy Robust PCA and a Classification Algorithm
Authors	Kurama, Onesfole; Luukka, Pasi; Collan, Mikael
Purpose	Classification is an essential part of credit analysis and bankruptcy prediction. A Financial institution can have a significant competitive advantage if it can separate good borrowers from a group of possible borrowers more accurately than its competitors. The problem is an actual business problem, failed credit decision-making in financial institutions may cause severe financial difficulties, while at the same time it hinders good business if money is not lend to good borrowers. This paper investigates combinations of three fuzzy robust principal component analysis algorithms and two different classifiers (similarity classifier and k-nearest neighbor classifier) to find out which combination gives the most accurate classification result. (Kurama et al., 2015).
Research methods	The data set used in this paper is the "Australian credit screening dataset" which can be downloaded from the UCL machine learning data repository (Lichman, 2013). Research method is a design science based research, the used algorithms are implemented and analyzed with the MATLAB software.
Conclusion and results	With parameter values set correctly, slightly above 80% classification accuracy can be obtained. There should be more tests done with other methods to be able to conclude whether these results are good or poor. This paper however shows clearly that choosing the parameter values has great impact on classification accuracy and this should be emphasized when implementing and using these systems. (Kurama et al., 2015).

The table above gathers the results for the first research question, which was: What previous academy research literature exists on similarity based classifiers and what are

the results of the said previous research? It can be concluded that there is not much research on similarity classifiers in business related context. Bankruptcy and credit scoring seem to be most applicable area for the similarity based classification. Also malware detections, social media and financial time series were used as application areas. Most of the research uses data from repositories available on the internet. It could also be that companies will not publish the work they have done, since it might affect their competitive edge if they do, and therefore a lot of the made research on the topic is not visible. But in general it is clear that there is a room for more research in this area.

4 BENCHMARKING CLASSIFIERS

4.1 Reference classifiers

There is a large variety of classifier algorithms available and many of those have some parameters, which makes the choices of how to classify a set of data items even more plentiful. Different classification algorithms have different approaches for learning. There is no one method that would be the best for all applications or one algorithm that would be best suited for all classification problems, even the best data scientist cannot tell which algorithm works best for a given problem until they make tests. Finding the right algorithm for a given problem is usually done in a heuristic way, by trial and error. (Mathworks, 2018).

Benchmarking classifiers is a heuristic way of investigating which classifier algorithm would be best suited for a given classification problem. Here the target is to find out whether the GOWA-variant of the similarity classifier is better suited for a business analytics context than algorithms from a set of classical classifiers. In the following the reference classifiers that are used for this benchmarking are presented shortly. The purpose is not to go into too much detail for each classifier, there is lot of information available for each classifier in relevant machine learning literature for anyone that is interested. The reference classifiers that are used in this thesis have been selected based on their common use among machine learning applications and literature, and their availability in Statistics and Machine Learning Toolbox in MATLAB, which is the software used in this thesis.

The different classification algorithms can be grouped in several ways. One way is to group the algorithms based on the method for which the class separation is based on. Figure 8 shows some of the commonly used classification algorithms grouped in this way.

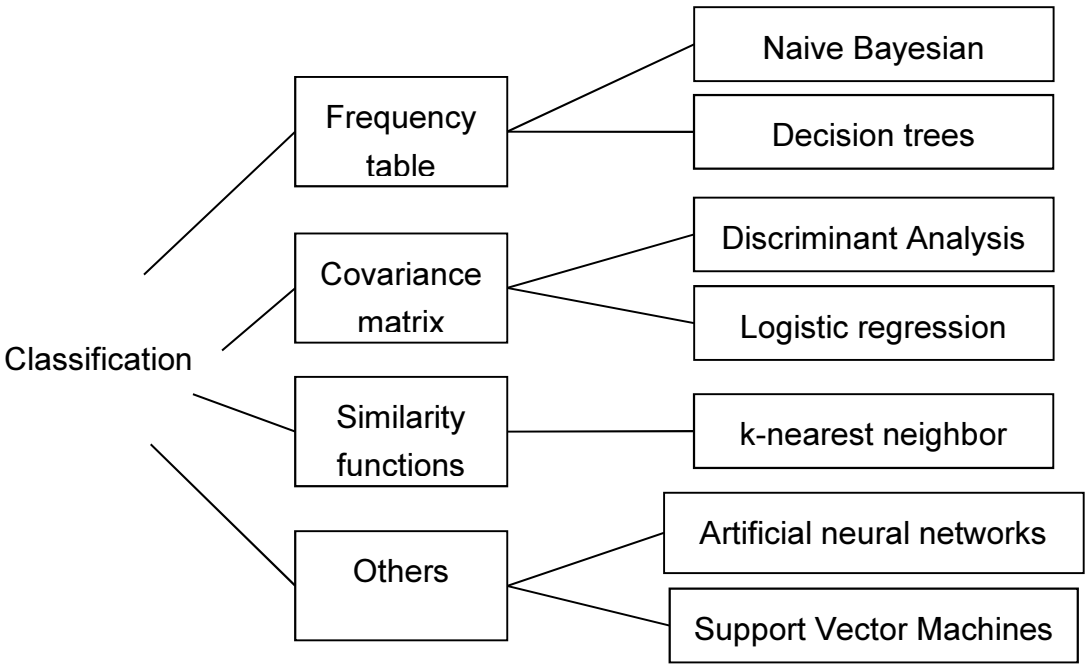

Figure 8. Classification methods

The four reference classifiers used in this thesis were selected so that the benchmarking is done with classifiers which are based on a different classification methodology. The selected four classifiers are introduced shortly below.

Decision tree

Decision tree is probably the easiest method to understand and interpret, even without any technical background. As the name implies, decision tree resemble a tree, or a branch, structure. An example of a decision tree is depicted in Figure 9. The root node is the starting point and in each decision node there are two choices, which divide the dataset according to given options. The decision nodes represent input variables, i.e., the predictors that are used for making the splits. The leaf nodes are the target classes, i.e., they represent the output variables. The decision tree structure is first learned from the training data set and after that each new record can be classified using the tree and its decision nodes. The prediction is made by walking through the decision nodes, starting from the root node, and ending up to one of the leaf nodes. (Le, 2018).

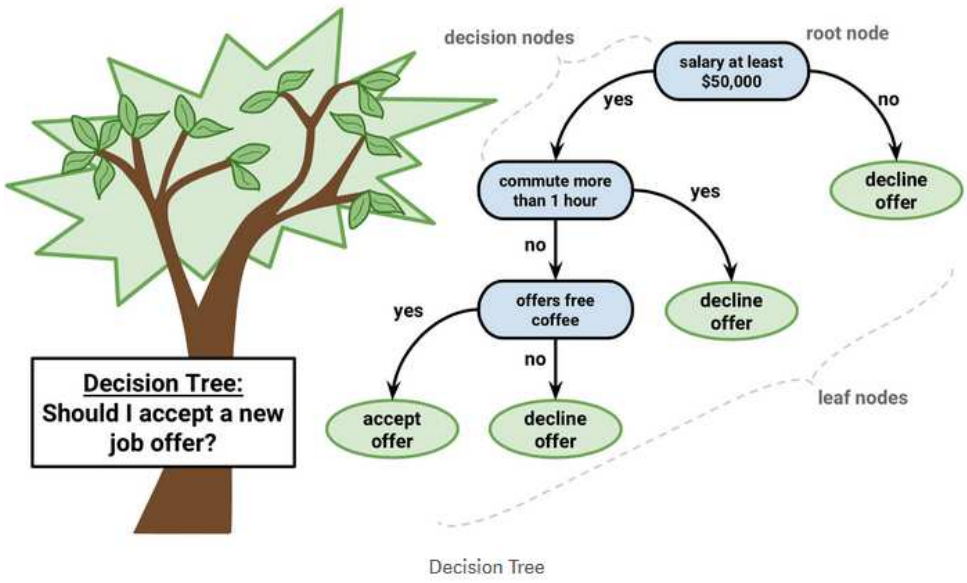

Figure 9. Example of decision tree (Le, 2018).

Decision tree tolerates errors and handle missing data well in a training data set and therefore do not require that much preparation of the data. Decision tree is very intuitive and the results are easily explainable. It can handle both numerical and categorical variables and does not have any requirement about linearity in the data. It is useful in exploring the data. Decision tree works well with discrete variables, while continues variables will result in instability. While the interpretation of the model is easy, it could be lacking on classification accuracy. Also there is a great risk of over-fitting and thus erroneous classification results, if the tree is grown "too deep". Decision tree could be used for instance by banks to classify loan applicants by determining, whether their will default on payments or by healthcare institutes to identify at-risk patients or disease trends. (DeZyre, 2018).

Discriminant Analysis

As a classifier which uses covariance matrixes as a basis for classification, a discriminant analysis classifier was chosen, since linear regression classifiers are limited to only binary classification problems, and the new-thyroid data used in this thesis, has three target classes. Discriminant analysis classifier uses linear or quadratic combinations of predictors. The target is to find such combinations that class variance between different classes is maximized relative to the variance within the class. This means that a mean value is calculated for each class and the variance across all classes and mean values of each class will be used as class boundaries. Predictions are then made by calculating a discriminative value for each class and the correct class label is selected based on the largest discriminative value.

Linear discriminant analysis can also be used to decrease attribute space, the target is then to project attributes to axes so that the projections will maximize the separation between classes, see Figure 10.

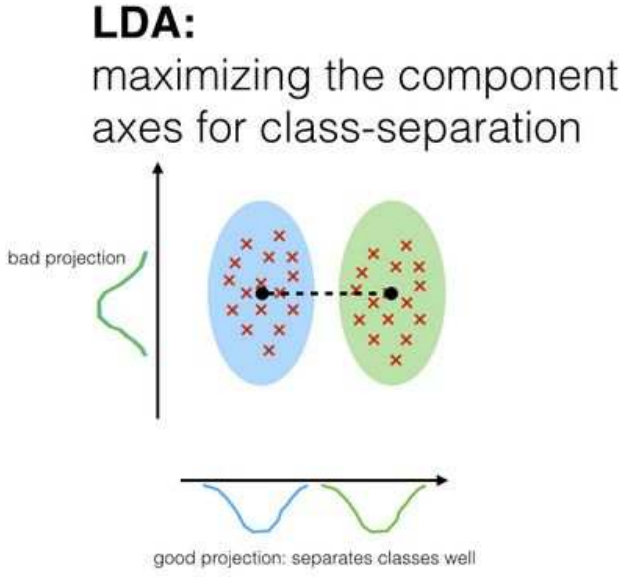

Figure 10. Linear discriminant analysis. (Raschkam, 2014).

The discriminant analysis assumes that all predictors have a Gaussian distribution. So if there are clear outliers, those should be removed before the classifier training is performed. (Le, 2018).

Support Vector Machine

Support Vector Machine (SVM) was first introduced by Cortes and Vapnik (1995) to be used for binary classification. The target is to find an optimal separating hyper-plane between two target classes, which will split the input variable space and maximize the separation margin between the classes' closest points. The points that lie on the boundaries of the separation margin are called support vectors. There might be some points of the vectors that lie on the "wrong" side of the separation margin, their influence can be reduced by giving them a lower weight. If a linear separator cannot be found, data points are projected into higher-dimensional space, where they become linearly separable. A programme able to execute all the required task that will generated the separation margin and thus make the classification is called a Support Vector Machine. (Meyer & Wien, 2001). Example of the support vector machine binary classification is depicted in Figure 11.

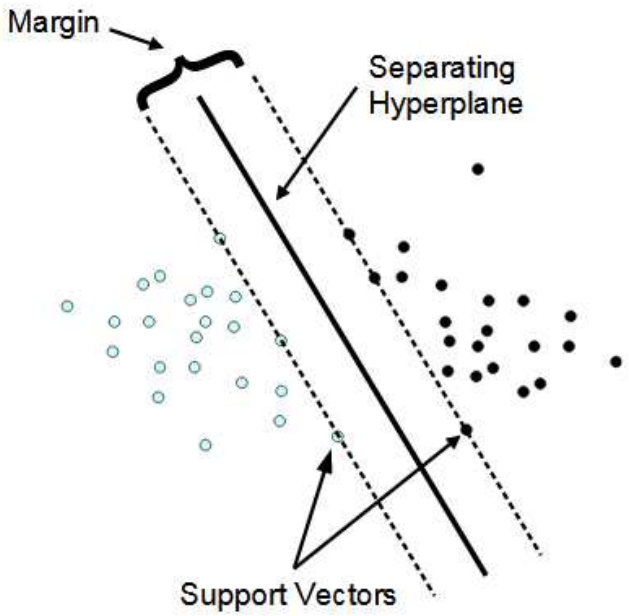

Figure 11 Example of Support Vector Machine (Meyer & Wien, 2001).

Support Vector Machine typically has good classification accuracy. The advantage of using SVM is that it does not make any strong assumptions about the data and it does not tend to over-fit the data. Support Vector Machine is commonly used for a stock market forecasting to make relative comparisons between different stocks. (DeZyre, 2018).

K-nearest neighbor

If it looks like a duck, walks like a duck and quacks like a duck, then it's a duck. This common phrase can be used to describe k-nearest neighbor (KNN) classifier to some extent. If a data item resembles its neighboring items, it is probably of the same type as the neighbors. The K-nearest neighbor method is based on identifying k items in the training dataset that are most similar to the data item to be classified. The target class is determined based on the neighboring data items, the new item is assigned to the class which is most predominant among the nearest neighbors. To assess which are the closest neighbors to the new item to be classified, we need to investigate the parameter space, i.e., the predictors and select the neighbors which predictors are closest to the predictors of that new item. There are different ways to determine the distance or the similarity between the data items or records, but Euclidean distance is the most commonly used method for that. Euclidean distance between two records $u = (u_1, u_2, \dots, u_n)$ and $v = (v_1, v_2, \dots, v_n)$ is calculated as:

$$E = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + \cdots + (u_n - v_n)^2} \quad (13)$$

K-nearest neighbor classification algorithm is simple to implement, it has a good tolerance against noisy data and it is effective for large training datasets. But the computation cost could be high, because the distances between each record need to be calculated to all training samples (Garg, 2018).

Example of the k-nearest neighbor classification is depicted in Figure 12.

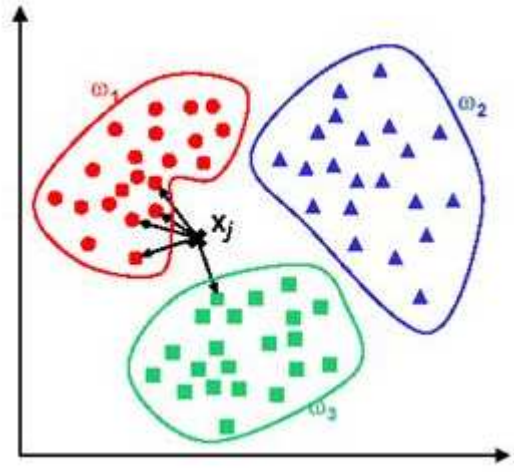

Figure 12. Example of k-nearest neighbor classification (Cengiz, 2017).

In Figure 12 there are three classes available and the goal is to find out a correct class label for the new record x_j. Euclidean distance is used for determining the nearest neighbors and the value of k=5, i.e., five closest neighbors are considered and by majority voting, class ω_1 is selected since four closest neighbors belong to that class.

4.2 Data sets

The benchmarking of classifiers is done with three different datasets, all of which can be obtained from the UCI Machine Learning database (Lichman, 2013). As a starting point for this thesis was the research work done by Kurama et al. (2017) with four different data sets, two of those data sets are also used in this thesis to get comparable data. In addition one new dataset, a credit default data set from the said database is used for comparing the GOWA-variant of the similarity classifier against the reference classifiers and to evaluate how well the GOWA-variant is suited for business-related problem solving. All used datasets are briefly introduced next.

Haberman's survivor data set

This dataset contains patient data from a study on survival of patients who has gone through a breast cancer surgery (Lichman, 2013). There are 306 instances in this data set with three numerical attributes. Attributes are the "age of patient" at the time of operation,

the “year the operation” was done (year -1900 in the dataset), and the number of “positive axillary nodes” detected. The class-attribute is the survival-status of the patient that indicates whether the patient has survived five years or longer after the surgery or died within five years.

New-Thyroid data set
This data set is one of the thyroid data sets found in the UCI Machine Learning data repository (2013). This data set contains 215 instances, which all have 5 attributes. The purpose of this data set is to investigate the thyroid symptoms of the patients. There are three target class attributes, normal (no thyroid symptoms), hypo (low functioning thyroid), and hyper (high functioning thyroid).

Default of Credit card clients
This data set contains customer data of a Taiwanese bank. The aim is to predict whether the customer is creditworthy or not. There are 23 explanatory variables in this data set. As an assumption, or a hypothesis, we can state that the similarity classifier is likely to give us better results for the classification that the traditional classifiers. The attributes of this data set are: amount of credit given, gender (1=male, 2=female), education (1=graduate school, 2=university, 3=high school, 4=others), marital status (1=married, 2=single, 3=others), age (year), history of last payments (6 attributes, -1=pay duly, 1-9 payments delay in months), amount of bill statements (6 attributes, status of 6 past months), amount of previous payments (6 attributes, status of 6 past months).

4.3 Evaluation of the selected classifiers

As we want to compare different classifiers, we need some criteria on how to evaluate the quality of each classifier. Two of the classification problems that are investigated in this thesis are binary problems, i.e., there are only two classes which the classifier is supposed to separate from one another. And there are also only two kind of mistakes the classifier could make; it could classify a positive item incorrectly as a negative item (called false negative = FN), or it could classify a negative item incorrectly as a positive item (called false positive = FP). The items for which the class is predicted correctly are called true positives (TP) and true negatives (TN). It is obvious that the classifier should aim to minimize both false negatives and false positives. But usually when one decreases one, one increases the other. And this is also the case with false positives and negatives. If the classifier would simply classify all items as positives there would not be any false negatives, but instead all negative items would be predicted incorrectly. Or if the classifier would predict all items as negatives, one would not get any false positives, but all positive

items would be incorrectly predicted as negative thus producing a lot of false negatives. Therefore the classifier must find a compromise between these two extremes, while trying to minimize both false negatives and false positives. (Berthold et al., 2010).

One way to visualize the trade-offs between benefit of the classifier (true positives) and cost (false positives) is to use a Receiving Operating Characteristics or ROC curve. ROC curve is one of most commonly used method to visualize the performance of a binary classifier. ROC curve was first used during World War II radar images to distinguish between enemy ships, friendly ships, or just noise. It was aimed at measuring the operating characteristics of the radar receiver which explains the name of this tool. Later it has been adapted by medical diagnostics and for the first time in machine learning applications by Spackman (1989) when he used ROC curve to compare and evaluate algorithms. (Gonçalves et al., 2014).

ROC is a plot of benefits, i.e., the sensitivity (true positive rate) against cost or noise (false positive rate, also noted as 1-specifity, i.e., predicting a true negative item as positive item). Some classifiers provide a probability of whether the predicted item belongs to the class of not. Usually class label is selected based on the highest probability. With binary problems, an item will be labeled as true if the probability of the true class is greater than 0.5. If we would select higher probability, say 0.8, we would result fewer false positives, but also more false negatives. ROC curve is used for illustrating different threshold values for the class separation probability. An example of a ROC curve is depicted in Figure 13. The diagonal line equals random guessing, if you pick a random positive sample, there is a fifty present chance the model predicts it as a positive sample and same for the negative samples. The area under the “curve” = the diagonal line, “AUC”, is 0.5, which equals to a fifty percent chance to predict the class correctly. An ideal classifier would have AUC = 1 (point (0, 1) in the Figure 13), which would mean that all new samples are predicted correctly. (Berthold et al., 2010).

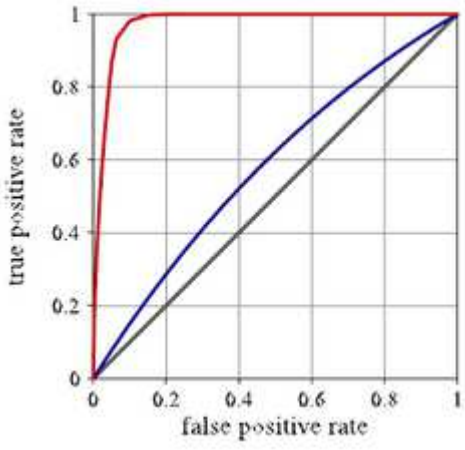

Figure 13. The ROC Curve. (Berthold et al., 2010)

ROC curve used a few numerical criteria to measure the performance of a classifiers. Hossin and Sulaiman (2015) present in their paper a set of quantitative indicators that can be used for evaluating classifier performance. The most obvious one is probably the overall classification accuracy; i.e., how well the classifier can predict a correct class. Accuracy can be calculated as follows:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

In other words, the overall accuracy measures the ratio of correct predictions over the total number of items measured. Hossin & Sulaiman (2015) also present many other quantitative indicators, such as the Error rate, Sensitivity, Specificity, Precision, Recall, F-Measure, or using different average values. The error rate (err) measures the ratio of incorrect predictions over the total number of instances evaluated.

$$err = \frac{FP + FN}{TP + FP + TN + FN}$$

Sensitivity (sn) measures the fraction of correctly classified positive patterns.

$$sn = \frac{TP}{TP + FN}$$

Specificity (sp) is similar to sensitivity, but instead of positive patterns, it measures the fraction of correctly classified negative patterns.

$$sp = \frac{TN}{TN + FP}$$

Precision (p) is a measurement criterion that determines the ratio of the correct positive predictions over the total number of positive predictions, i.e., how many of the predicted true elements are actually true.

$$p = \frac{TP}{TP + FP}$$

Recall (r) measures the fraction of actual true elements of the correctly predicted instances.

$$r = \frac{TP}{TP + TN}$$

F-Measure is a harmonic mean between recall and precision values.

$$FM = \frac{2 * p * r}{p + r}$$

It depends on the features of the data how well each of these quantitative evaluation methods work. None of them is an absolute best measurement criterion, there are some cases where a particular evaluation criterion may result in too positive outcome and thus some other criteria should be used instead. For instance accuracy is not a good evaluation criteria if the dataset is heavily imbalanced, such as when classifier tries to detect fraudulent transactions from legitimate ones. Presumably most transactions are legitimate, so the classifier can classify all transactions as legitimate and achieve 99% accuracy and still be a totally useless classifier. I.e., accuracy is a good evaluation criteria

if the target classes are somewhat in balance is the evaluated dataset. Precision does not give much useful information, if there are only few positive elements in the evaluated dataset, but it fits well in cases where each try has a significant cost, but missing a chance does not matter much – an example of such a case could be a recruitment process. Recall could give too good evaluation results if the dataset is biased towards positive items. It is easy to predict true values if most the predicted values are true. But this criteria is valuable if missing a positive item will cost a lot, like for instance a cancer prediction or some other medical diagnostic case. (Thoma, 2018).

Confusion matrix is another popular graphical indicator tool in addition to the ROC curve, which suites for binary classification problems well and it is also used in this thesis. Confusion matrix, which can also be called an error matrix, is a specific table format which summarizes the performance of a classifier. Actual classes are represented in rows of the matrix and predicted classes in columns, or vice versa. From the table is it easy to see whether the classifier can distinguish the classes from another or does the classifier confuse classes. (Sokolova & Lapalme, 2009).

The error rate is also easily visible from the table, since both correctly predicted class items and falsely predicted class items are presented in the table, see Table 2.

Actual data class	Classified as positive	Classified as negative
positive	TP (true positive)	FN (false negative)
negative	FP (false positive)	TN (true negative)

Table 2. Confusion matrix (Sokolova & Lapalme, 2009).

Table 3 shows an example of a confusion matrix. Say that we have one thousand emails, of which 500 and real emails and 500 are spam. The confusion matrix shows how well the classifier can separate spam email from real emails.

Actual class	predicted as real email	predicted as spam
real email	470	30
spam	80	420

predicted as real email	predicted as spam
94%	6%
16%	84%

Table 3. Example of a confusion matrix. Predicted classes shown both in number of observation and percentage format.

Other metrics to represent the classifier performance can also be calculated from the figures presented in the confusion chart. For instance accuracy for the spam classifier would be:

$$acc = \frac{TP + TN}{TP + TN + FP + FN} = \frac{470 + 420}{470 + 420 + 30 + 80} = 89\%$$

For this thesis a fitness function is created for each case, which would combine selected measurement criteria. A suitable weight is set for each criteria and an overall performance criterion is produced as a result of this fitness function. The fitness functions will be presented in more detail with the case results in chapter 4.5.

For the most classification problems, one does not know beforehand which classifier algorithm is the best for the current problem. One needs to iterate between training and testing different classifiers and fine tune the parameters before one can select the best model for the current problem. After one has selected the best model, one still wants to test it with some data that was not used for training the model to get more reliable results. In this thesis different classifiers are compared against each other, but all of them are also tested at the end to get the final results, i.e., no actual model selection is performed.

Before data is used for training a classifiers, it is usually partitioned into two sets, classification model selection data set and test set. Partitioning is performed randomly, but the same partition should be used for training and cross-validating each classifiers to get more comparable results. A part of the data is left out for testing and the other part of the data is used for training the classifiers and selecting most suitable parameters. The part that is left out as a test set mimics the real-world new data. That is, if the classifier would be used in real-world with new data that has not been seen before. The model selection part of the original data is further partitioned into k-folds to be able to perform cross-validation for the model. Cross-validation means that training and validation is performed several times (k times) and resulting classification error is calculated as a mean value of the individual iterations. For a larger data set one could use more folds, but 10-folds are quite common choice. The 10-fold cross validation scheme in general is depicted in Figure 14.

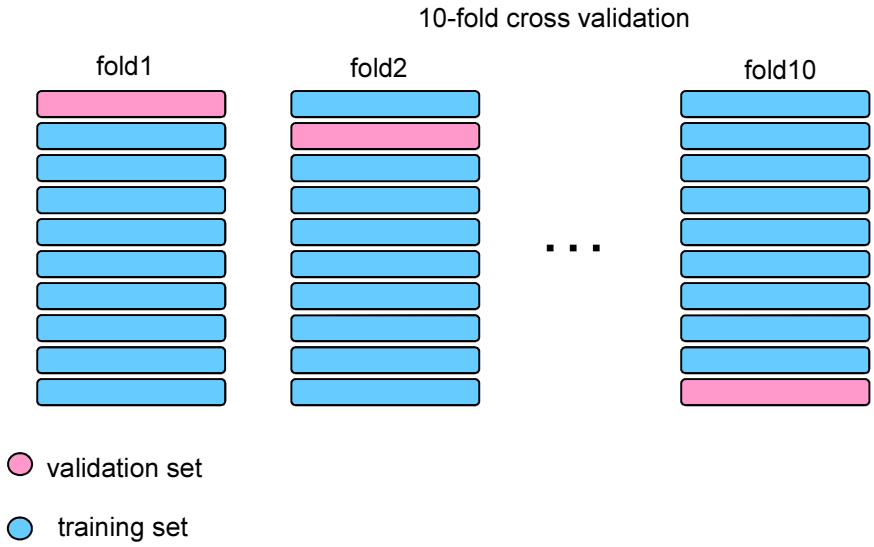

Figure 14. 10-fold cross validation.

The same 10-fold partitioning should be used for all the classifiers to get better comparable results, and this is how the cross-validation is performed in this thesis.

4.4 Test setup

The test setup for the evaluation of classifiers is illustrated in Figure 15. The data from each of the three data sets are fed into separate classifier algorithms. As a result each classifier produces a set of evaluation metrics (results). The values from those metrics are used in calculating the value of the selected fitness functions for each case. The resulting values from these fitness functions are used for generating the final ranking of the classifiers.

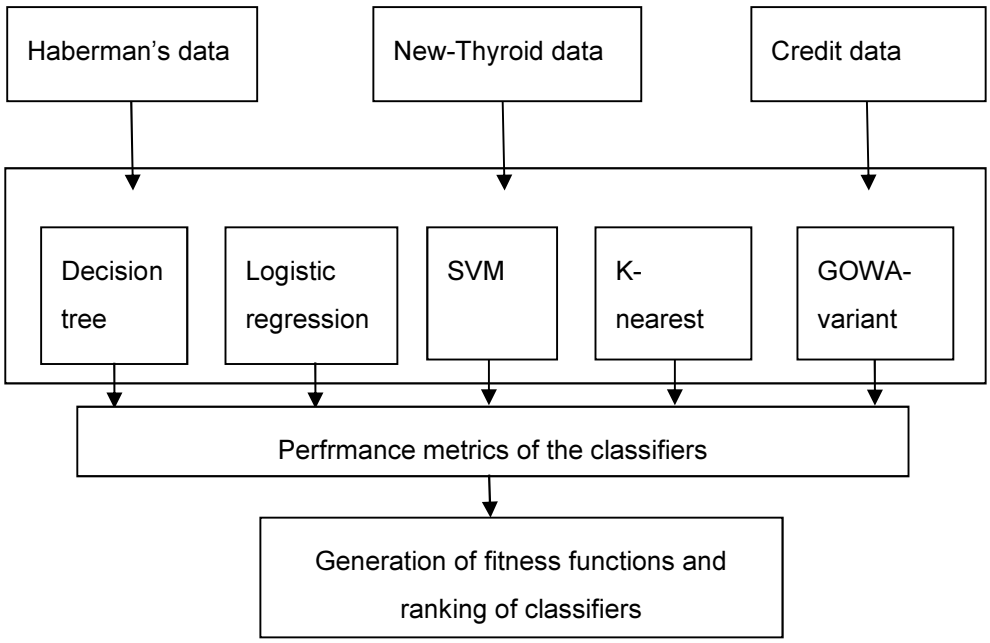

Figure 15. Test setup for classifier evaluation.

Similarity classifier with the OWA operator toolbox in MATLAB is freely available and contains most of functions that are needed for the similarity classifier used in this thesis. There are a few modifications made for the functions in that toolbox to make the GOWA-variant of the similarity classifier. The modified scripts were provided for this thesis from the original authors.

4.5 Results

4.5.1 Comparing classifiers with Haberman's survival data

First reference data is Haberman's survival data. This data set contains information of patients who have undergone breast cancer surgery. Predictor variables are the age of the patient, the year the surgery was made and the number of positive lymph nodes detected. According to Breastcancer.org web site (2018), doctor usually removes one or more of the underarm lymph nodes before or during the surgery, and then a pathologist examines how many of those nodes contain cancer cells, i.e., how many of the nodes are detected as positive nodes. First, the data set is explored in general and visualized in Figure 16. The visualization in Figure 16 shows that none of the predictors is alone able to separate the survival status by themselves. The survival status is spread quite evenly when plotted against each predictor separately in the left side of the figure.

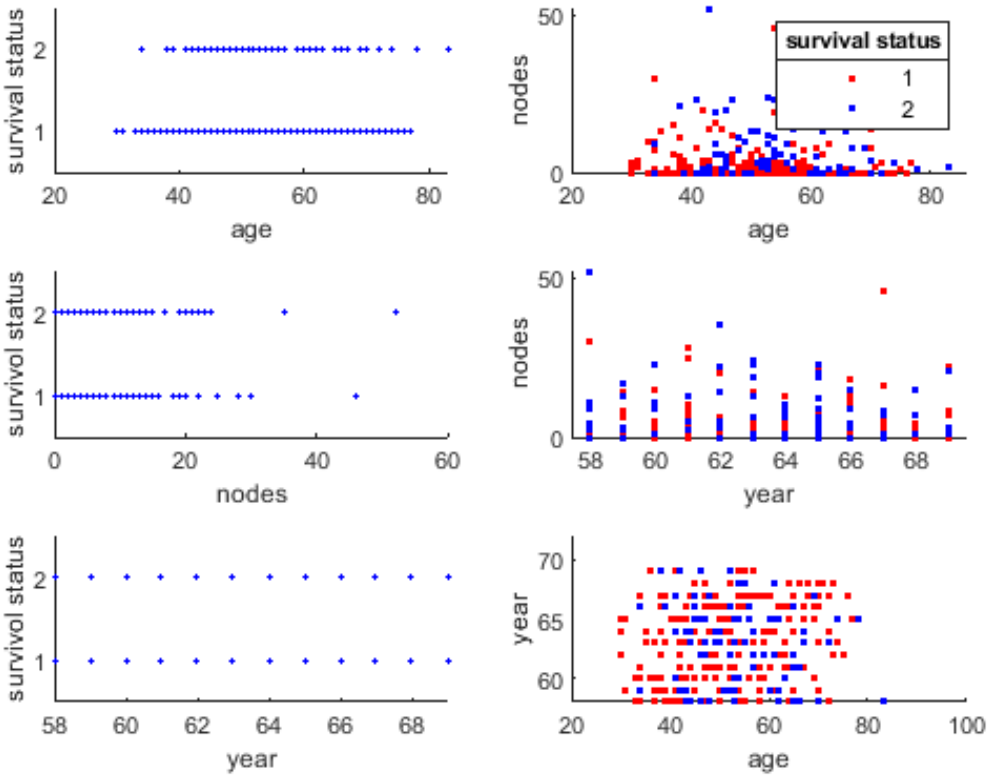

Figure 16. Haberman's survival data.

On the right side of the Figure 16 each combinations of two predictors are depicted and grouped by survival status. Red dots equal patients who have survived 5 years or longer after the surgery and blue dots equals the patients who have died within 5 years after the surgery. All the three subplots indicates that two target classed (survived and not survived) are not easily separated from each other. This will lead to the assumption that it is not easy to construct a classifier which would predict the target classes with good accuracy. The Haberman's survival data contains 306 observations, so it is a fairly small data set. This typically also has an effect on classification accuracy. As noted already, there are two target classes; 1 = survived patients and 2 = not survived patients. Out of 306 instances, 225 are labeled "survived" and 81 as "died" within five years after the surgery. These numbers are calculated as follows:

```
observation_count = height(haberman)
%find out the separate target classes
testClasses = haberman.class_label;
[classes,~,idx] = unique(testClasses);
classes
%calculate how any instances there are in each class
nCount = accumarray(idx(:),1)
```

There are no missing items and no clear outliers so the there is no need to process the data before it can be used for input data to the selected classifiers. If a classifier would

put label ‘1’ to all samples, it would gain 225/306=73.5% accuracy. So to be useful, a classifier should have better than a 73.5% accuracy, and if accuracy is close to that, it should also be checked that is does not label most of the samples as negative.

Before the Haberman’s survival data is used for training any of the classifiers, it is partitioned into two sets, classification model selection data set and test set. Partitioning is performed randomly, but the same partition is used for each of the classifiers.

```
rng(1);
c1 = cvpartition(thyroid.class_label,'HoldOut', 0.3);
testData = thyroid(c1.test, :);
test_predictors = testData(:, predictorNames);
ClassData = thyroid(c1.training, :);
predictors = ClassData(:, predictorNames);
```

30% of the data is left out for testing (testData in a script above) and 70% of the data is used for training the classifiers and selecting most suitable parameters (ClassData in a script above). The 30% test set mimics the real-world data. That is if the classifier would be used in real world with new data that has not been seen before. 70% of the original data is further partitioned into 10-folds to be able to perform cross validation for the model. The same 10-fold partitioning is used for all the classifiers to get better comparable results.

```
rng(1);
cv = cvpartition(ClassData.class_label,'KFold',10);
```

There is a Classification Learner App in MATLAB, which can be used to easily compare a set of classifiers. There are a set of different classifiers already available in that Learner App and there are also some parameters that can be changed before the selected classifier is trained. Results, such as the classification accuracy, are visible in that same application. The Classification Learner App was used as a starting point and as a simple way to get to know the data and to get some rough figures about what the classification accuracy should be. Classifier algorithms that are available in the Statistics and Machine Learning Toolbox for MATLAB, was used to train and optimize the models and to perform some of the quality measurements. The algorithms that were used are as follows:

- Decision Tree - `fitctree`
- Linear Discriminant Analysis - `fitcdiscr`
- Support Vector Machines - `fitcsvm`
- K-nearest neighbors - `fitcknn`

Each of these classifier algorithms has various parameters that can be tuned to have better classification results. In this thesis a fitness function is used, instead of only using classification accuracy, to evaluate the quality of the classifiers. Fitness function combines a few quality criteria into a one figure. For the Haberman’s survival data, the following fitness function was used:

$$f_{fitness} = 0.25 * sn + 0.25 * sp + 0.5 * ACC$$

There are three different measurement criteria in that fitness function, classification accuracy has the biggest weight 0.5 and sensitivity and specificity both have weights 0.25. Of course one might argue that it is more important to find the positive instances (in this case the patients who did not survive) than the negative ones. But since we do not have enough domain knowledge to make that selection, in this thesis the two are weighted equally. The fitness function will be used in each classifier to find out optimal parameters and also to make the final ranking for the classifiers in this classification problem.

For each of the four classifiers, a slightly different ways are used to find out the optimal parameters. But the main idea remains to same; classifiers are trained and cross-validated several times and each time a fitness function value is calculated. After the runs, the maximum value of the fitness function is determined and the parameters that gave that particular value were selected to train the model again and to produce the final cross-validation results. The MATLAB codes that were used to optimize and cross-validate the classifiers can be found from the appendix. The relevant codes are identified as knn_classifier.m, tree_classifier.m, svm_classifier.m, discriminant_classifier.m and gowa_classifier.m.

To find out the optimal parameter values for KNN classifier, the fit function was first run with OptimizeHyperparameters set on, see below:

```
rng(1);
Mdl = fitcknn(predictors, ClassData.status, 'OptimizeHyperparameters',
      'auto', 'HyperparameterOptimizationOptions',
     struct('AcquisitionFunctionName', 'expected-improvement-plus'));
```

See Figure 17 for the results of the Hyperparameter optimization run.

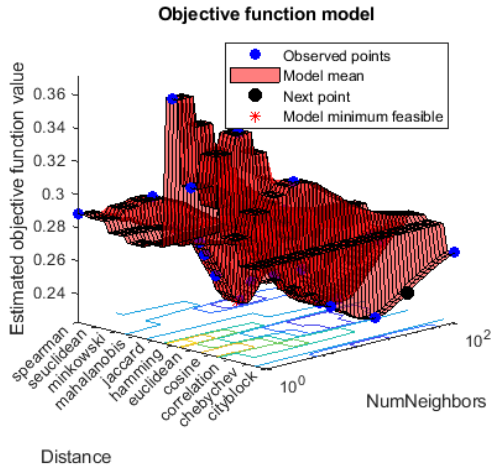

Figure 17. Parameter optimization for knn classifier for Haberman's data.

In addition to the parameter values resulting from hyperparameter optimization, K-nearest neighbor classifier was also trained and cross-validated with 'cosine', 'euclidean',

'jaccard', 'chebychev' and 'seuclidean' distances and with 10 to 30 neighbors. The parameters which resulted in the best fitness function value was selected to be used for the final model. See the knn_classifier.m script from the appendix for details.

For the decision tree classifier parameter-optimization, the Classification Learner App was first used to select the split-criteria for the tree. The application suggested that Gini's diversity index would be the best split-criteria and thus that criterion was selected. The tree_classifier function alters two other parameters, maximum number of splits with values ranging from 2 to 14 and minimum leaf size with values ranging from 12 to 30. See tree_classifier.m script from the appendix for details.

For the Support Vector Machine classifier two options for the parameter optimization were selected. Both of these optimization methods can be found from the MATLAB Statistics and Machine Learning Toolbox help pages for SVM classifier training and fine tuning parameters. First, a hyperparameter optimization run was performed in a similar way it is done for the KNN classifier optimization. See below:

```
opts = struct('Optimizer', 'bayesopt', 'ShowPlots', true, …
       'CVPartition', cv, 'AcquisitionFunctionName', …
       'expected-improvement-plus');
rng(1);
svmmod = fitcsvm(predictors, ClassData.status, 'KernelFunction',…
        'rbf',    'OptimizeHyperparameters', 'auto',…
        'HyperparameterOptimizationOptions', opts);
```

BoxConstraint and KernelScale values that resulted were then used for training and cross-validation of the model. The second option for parameter optimization was first to run the model with KernelScale set to 'auto' mode.

```
svmmod2 = fitcsvm(predictors, ClassData.status, ...
                  'KernelFunction', 'gaussian', ...
                  'PolynomialOrder', [], 'KernelScale', 'auto', ...
                  'BoxConstraint', 1, ...
                  'Standardize', true, 'ClassNames', [1; 2]);
```

This resulted in a KernelScale value which was used as a basis for the further runs. In those runs the KernelScale value BoxConstaint values were adjusted by increasing them by a factor 10 in each round for 11 times. The scale factors were selected as advised in MATLAB help pages. See svm_classifiers.m script from the appendix for details.

For the discriminant analysis classifier all available discriminant types were tested (linear', 'pseudolinear', 'diaglinear', 'quadratic', 'pseudoquadratic', 'diagquadratic'). FillCoeffs parameter was tested with both "on" and "off" and also for the gamma value both, 0 and 1 values were tested. See discriminant_classifier.m MATLAB script from the appendix for details on classifier training and optimization.

Confusion matrixes for all cross-validated classifiers for Haberman's survival data are depicted in Figure 18.

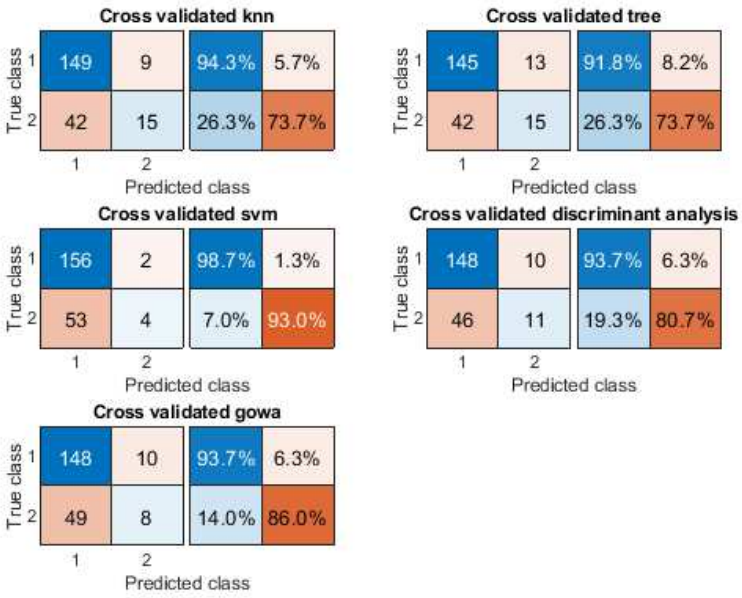

Figure 18. Confusion charts for cross-validated Haberman's survival data.

From the confusion matrixes it can be easily seen, that the models could predict the patients who survived five years or longer after the surgery quite well, but did a lousy job with the patients that did not survive.

The GOWA-variant of the similarity classifier was trained with same data as the reference classifiers, but since there are no build-in feature for the k-folds and cross-validation, the 10 folds were calculated separately and then the mean for the classification accuracy and fitness function was calculated for these 10 folds.

```
for k = 1:cv.NumTestSets
    trIdx = cv.training(k);
    teIdx = cv.test(k);
    datalearn = table2array(ClassData(trIdx, :));
    datatest  = table2array(ClassData(teIdx, :));
   [datatest, tc_sizes, cstart_idx]  = init_data(datatest,v,c);
   [datalearn, lc_sizes, cstart_idx] = init_data(datalearn,v,c);
    ideal_vec = idealvectors(datalearn, y);
   [fitness(k), class, Simil] = calcfitness(datatest, ideal_vec, y);
   [acctmp, sn, sp] = evaluate_classifier(class, datatest(:,end), …
                      classes, 0, '');
    fit(k) = calc_fit(fitness(k), sn, sp, fitness_weights);
end
meanfit(j,i) = mean(fit);
```

Init_data function above will sort the samples so that they are ordered by classes. Ideal vectors are calculated for each training data set. Similarities are then calculated between the test data units and the generated ideal vectors. The calcfit function returns the predicted class labels and the classification accuracy together with the similarity values. Evaluate_classifier function will calculate sensitivity and specificity figures for the predicted results and the fitness function value is calculated based on those figures the

same way as it is done with the reference classifiers. The 10-fold cross-validation results for the GOWA-variant of the similarity classifier can be seen from the confusion matrix from Figure 18.

The above mentioned procedure for the GOWA-variant of the similarity classifier was repeated several times with different parameters. Parameter α relates to the quantifiers, see equations 6-9 in chapter 3.3 for details, whereas parameter p relates to equivalence (equation 10 in chapter 3.3) and parameter λ to the similarity measure (equation 11 in chapter 3.3). Parameter α varies between [0.25:0.25:5] and p between [0.1:0.25:4]. Mean classification accuracies with these parameter value ranges are plotted in Figure 19. From the figure it can be seen, that the maximum classification accuracy is gained in the middle of the grid, so it can be claimed that selected parameter ranges are correct and actual maximum value for the accuracy is achieved within those ranges.

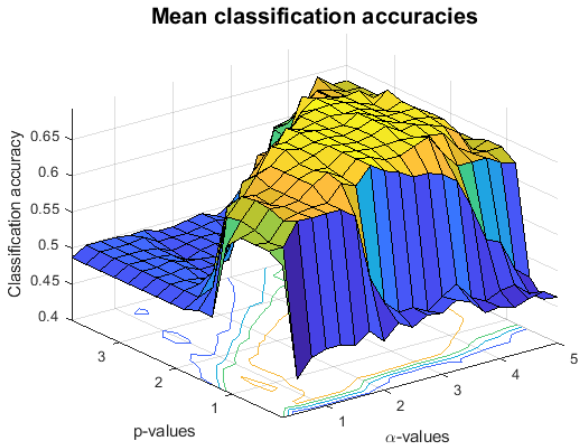

Figure 19. Parameter optimization for Haberman's survival data.

Different values were also investigated for λ. Different lambda values from the range [0.1:0.1:5] were tested. Classification accuracies, sn and sp values are plotted in Figure 20.

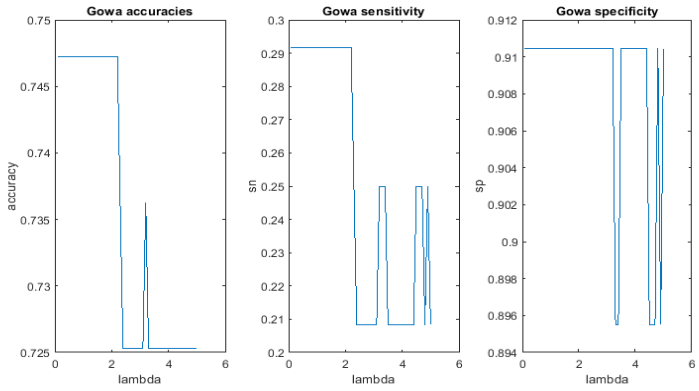

Figure 20. GOWA results with different lambda values.

From the picture above, it can be seen that best performance was gained with many different λ values, of which the value 2.0 was selected. It was observed that the choice of a quantifier did not have much effect on classifier results, so the basic RIM quantifier which gave best results was used for all the runs with different parameters.

After the all five classifiers were trained and parameters optimized with the model selection part of data (70% of the original data), they were all tested with the test data set (30% of the original data), i.e., with the data that the models have not seen before. The results for test data are gathered in table format in Figure 21. All the measurement criteria values that were used for the final fitness function generation are also visible in that table. Also the cross-validation accuracy figures are depicted there for reference.

	sp	sn	accuracy	cross_val_acc	fitness
knn	0.9403	0.20833	0.74725	0.76279	0.66078
tree	0.89552	0.29167	0.73626	0.74419	0.66493
svm	0.97015	0	0.71429	0.74419	0.59968
discriminant	0.9403	0.29167	0.76923	0.73953	0.69261
gowa	0.95522	0.29167	0.78022	0.72558	0.70183

Figure 21. Haberman's survival data results.

If we look at fitness function results, it can be seen that the GOWA-variant of the similarity classifier had the best performance. GOWA-variant also has the best classification accuracy for the test data. The best cross-validation classification accuracy figures are received with k-nearest neighbor classifier. Confusion charts for the cross-validated models were already illustrated in Figure 18, but they are depicted again for the test data set in Figure 22.

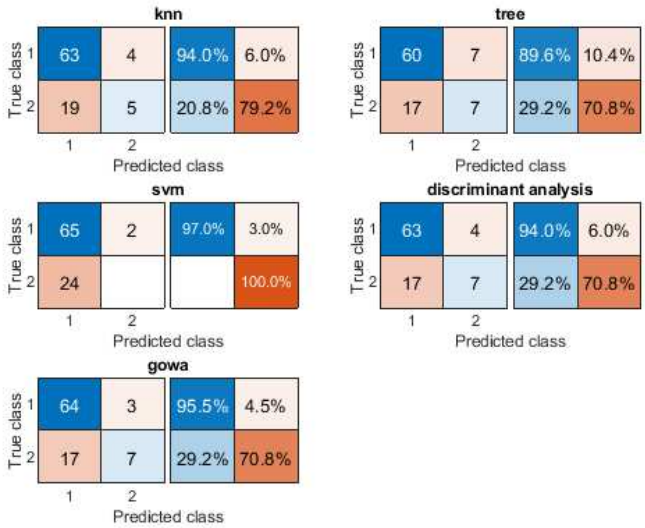

Figure 22. Confusion charts for Haberman's survival test data.

Confusion charts for cross-validation data and test data should be similar as the data partitioning was random and both sets (70% and 30%) should describe the data the same way. But as the data set was quite small, there are clear differences. The true accuracy figures are probably something between these two figures. If there would have been more instances, the confusion charts for model selection data set and for the test set would have probably been closer to each other.

ROC curves are not used for the classifier ranking for this thesis, but they are depicted for reference for the test data for all the four reference classifiers in Figure 23.

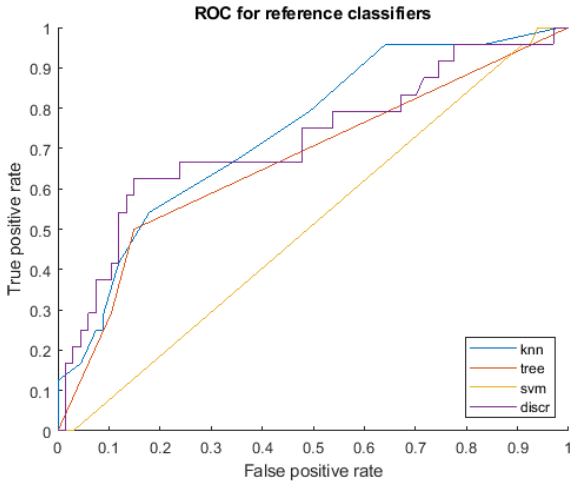

Figure 23. ROC curves for the reference classifiers for Haberman's survival data.

According to the ROC curve, there is no clear winner for this dataset, but svm is clearly the worst performing classifier for this test data. The ROC curve is not depicted for the GOWA-variant of the similarity classifier, since calculating it falls outside the scope for this thesis.

It was observed with the second data set (New-thyroid data), that the results varied when the original data split for the model selection data and test data was made with different random seeds. Since Haberman's survival data set and New-thyroid data set are both rather small, the similar variations are probable also for the Haberman's survival data set. That is why the whole procedure described above was repeated six times with different random seeds and mean values are calculated for those repeated runs to get more robust results. The results presented earlier in this chapter are reported from the last run.

```
for r=1:6
   rng(r*16);
   c1 = cvpartition(haberman.class_label,'HoldOut', 0.3);
   ...
```

The results from six consecutive runs and mean values for the fitness function and for the classification accuracy are depicted in table format in Figure 24.

	knn	tree	svm	discriminant	gowa
fit1	0.647	0.645	0.618	0.647	0.653
fit2	0.641	0.715	0.636	0.679	0.709
fit3	0.677	0.62	0.626	0.663	0.662
fit4	0.686	0.726	0.672	0.686	0.688
fit5	0.613	0.67	0.62	0.657	0.698
fit6	0.661	0.665	0.6	0.693	0.702
mean_fit	0.654	0.673	0.629	0.671	0.685
mean_accuracy	0.7473	0.7491	0.7234	0.7619	0.7381

Figure 24. Haberman's data results with different random seeds.

From the table above it can be seen that the GOWA-variant of the similarity classifier performed the best. It does not have the best classification accuracy, but it has the best fitness function value, which was the main performance measurement criteria in this thesis. It seems to be, that with small data set it is hard to make a robust classifier. This supports the claimed stated earlier in this thesis, that there need to be enough data available to make a good predictive model.

4.5.2 Comparing classifiers with New-Thyroid data

The second data set contains information about thyroid disease. This data set differs from the Haberman's survival data, since it has three target classes instead of two. So this is not a binary classification problem as such, but it can be divided into several sub problems that are binary. This will be done for the Support Vector Machine classifier, since SVM can only be used with binary problems. The class attribute values are 0 (=euthyroidism), 1 (=hyperthyroidism), and 2 (=hypothyroidism). Euthyroidism refers to a patient who has a normally operating thyroid. Hypothyroidism means that the thyroid gland does not make as much thyroid hormone as the body needs, i.e., the thyroid is underactive. Hyperthyroidism means that there is too much thyroid hormone, i.e., the thyroid is overactive. The predictive variables are different hormone levels measured from patients' blood samples. In Figure 25 there are several scatter plots, which give a general view about how variables can separate the class labels.

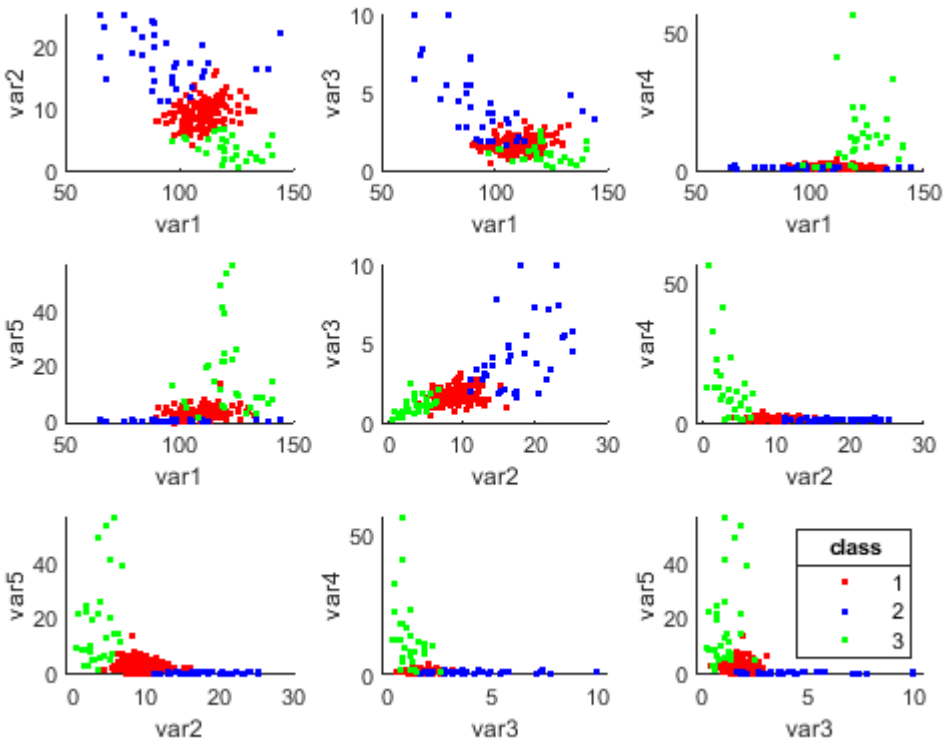

Figure 25. New-Thyroid data.

From the scatter plots it can be seen that already two variables separate the three classes quite well. For instance, in subplots where there are variables one and two or variables two and three, each three classes are clearly seen as separate groups. So the assumption is that also the classifiers should be able to categorize the data quite well with good accuracy.

There are 215 observations is this data set, so it is a rather small data set. 150 instances are labeled as patients who have normally operating thyroid, 35 observations are labeled as hyper, and 30 as hypo-operating thyroid. These numbers were calculated as follows:

```
observation_count = height(thyroid)
%find out the separate target classes
testClasses = thyroid.class_label;
[classes,~,idx] = unique(testClasses);
classes
%calculate how any instances there are in each class
nCount = accumarray(idx(:),1)
```

If the classifier would put a normal operating label (’0’) to all samples, it would achieve 150/210=71.4% classification accuracy, so the classification accuracy needs to be significantly better than that. We won’t use the same fitness function as we used for Haberman’s survival data, since we have three target classes now and there is no single positive and negative classes (sn and sp figures are normally calculated as a fractions of

correctly classified positive and negative instances). For the New-thyroid data we use the following fitness function to assess the classifier performance:

$$f_{fitness} = 0.55 * ACC + 0.15 * sn1 + 0.15 * sn2 + 0.15 * sn3$$

where ACC is accuracy, sn1 is sensitivity for class '0' (normal), sn2 is sensitivity for class '1' (hyper) and sn3 is sensitivity for class '2' (hypo). Sensitivity for each class is calculated as a fraction of correctly classified class items, see below:

$$sn1 = \frac{TP(0)}{TP(0) + FN(10) + FN(20)}$$

where TP(0) equals correctly classified class '0' items, FN(10) means items for which the true class is '0', but they are classified as '1' and FN(20) means items for which the true class is '0', but they are classified as '2'. In the fitness function, accuracy has the biggest weight, and the sensitivities for each class have been given an equal weight. It could be argued that it is more important to find out who are the patients whose thyroid is not operating normally, but as for this thesis we do not have decent domain knowledge for this issue, equal weights are selected.

The data set was partitioned in the same way as the Haberman's survival data. The test data set was separated first (30% of data). The rest (70%) was used for training, optimization, and cross-validation. Again the same partition was used for all the classifiers. For the cross-validation only 4-folds were used, since the data set was fairly small, and with more folds, the test data sets would have been quite small, which could cause variations in results. As the data set is small, it was observed that there were also variations in results, if the random seed was changed when the original data split (30/70) to model selection and test data was generated. Therefore the training and testing of the model was run six times with different random seeds for the initial data split for all the classifiers to get more robust results. Results of the runs with different random seeds are gathered at the end of this chapter.

```
for r=1:6
rng(r*45);
c1 = cvpartition(thyroid.class_label,'HoldOut', 0.3);
testData = thyroid(c1.test, :);
test_predictors = testData(:, predictorNames);
ClassData = thyroid(c1.training, :);
predictors = ClassData(:, predictorNames);

. . . % all the classifier training and testing etc.

end
```

K-nearest neighbor hyperparameter optimization was run with each six random seeds, and it gave results for optimum distance measure and the number of neighbors. In each

round knn was also trained with selected distance measures: {'cosine', 'euclidean', 'jaccard', 'chebychev', 'seuclidean'} and with 2…25 neighbors. The parameters which gave the best fitness function value were selected for each round and the final model was generated with those values. The results that are presented here (confusion matrixes etc.) are all done with the last of the six rounds. In that round the optimal parameter values were 'correlation' for distance and 6 neighbors.

Decision tree classifier and discriminant analysis classifier were trained and optimized the same way they were with the Haberman's survival data. But for the SVM classifier a different algorithm was needed, since the (MATLAB) fit function that was used for Haberman's survival data can only be used for binary problems. SVM was trained with fitcecoc function, which uses a templateSVM function as a basis. Three target class problem is divided into three binary problems and the results are then combined. The two functions (templateSVM and fitcecoc), that are available in the Statistics and Machine Learning Toolbox in MATLAB, will do that automatically. See the example script below:

```
template = templateSVM('KernelFunction', 'gaussian', ...
                       'PolynomialOrder', [], ...
                       'KernelScale', 'auto' , ...
                       'BoxConstraint', 1, ...
                       'Standardize', true);
classificationSVM2 = fitcecoc(predictors, ClassData.class_label, ...
                     'Learners', template);
```

The code above was run with three different choices for the KernelFunctions ('gaussian', 'linear', polynomial'). KernelScale was set to 'auto' should find optimal values for KernelScale. Hyperparameter optimization was also used to find out the optimal Learning parameter, KernelScale value and BoxConstraint value for each run. Fitness function was used for selecting the best parameter options, which were then used to build the final model. See the multisvm_classifier.m script from the appendix for details.

For the GOWA-variant of the similarity classifier, we used the same approach that was used for the Haberman's survival data. Parameter p was tested with range [1.0:0.25:8] and α parameter with range [0.5:0.25:8]. Also the effect of different lambda values from within the range [0.1:0.1:5] were simulated. The performance figures for different lambda values are depicted in Figure 26. From the figure it can be seen that values close to 2.0 gave the best performance figures, thus the value 2.0 was selected and the reported results were simulated with that value. Also with this data set, only basic RIM quantifier was used for weight generation.

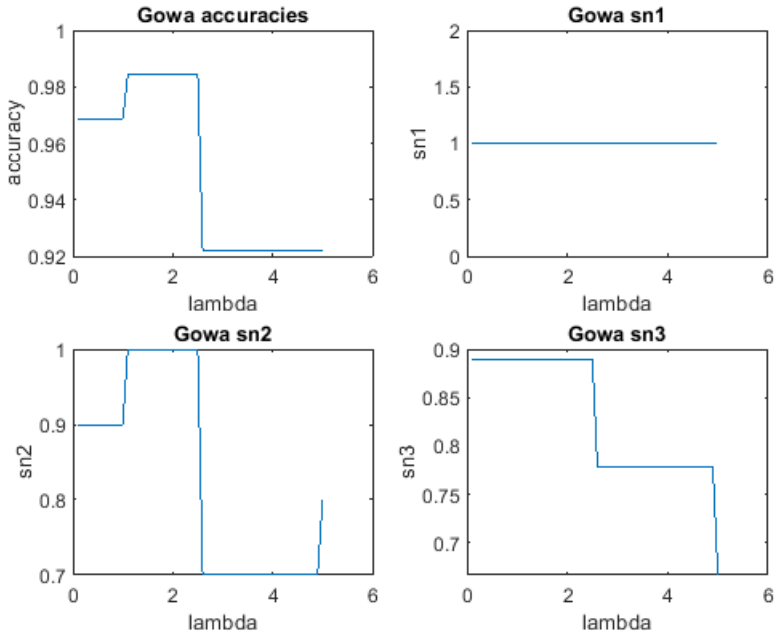

Figure 26. GOWA-variant performance with different lambda values.

The mean classification accuracies with New-thyroid data for the GOWA-variant of the similarity classifier are depicted in Figure 27. The maximum accuracies are gained in the middle of the grid, thus the used range for parameters p and α can be said to be correct.

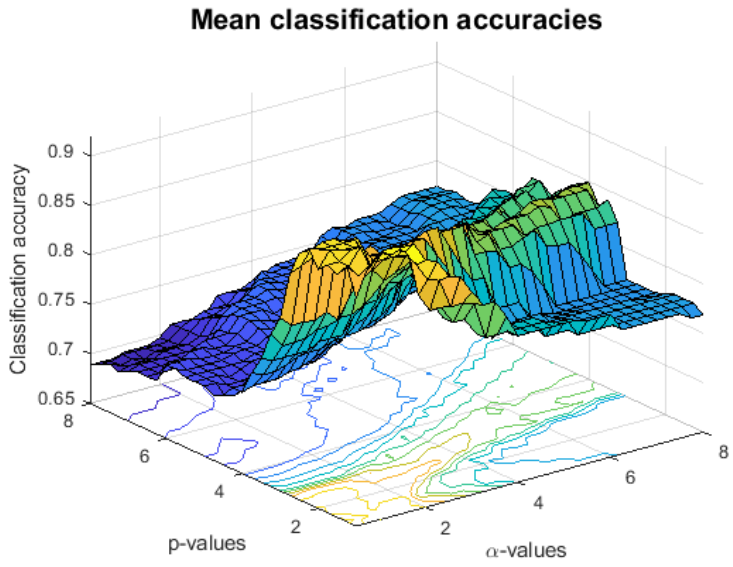

Figure 27. Mean classification accuracies for GOWA-variant with New-thyroid data.

As already mentioned before, all the five used classifiers were optimized, trained and tested six times, since it was noted that the random seed used for initial data separation for model selection and test data has an effect for classifier performance. The cross-validation results for all the classifiers are reported with the last round, were random seed

number was 270. (Normative selection was used for random seeds, value r*45 was used, where r is an integer from between 1…6.). Confusion charts for cross-validated models are illustrated in Figure 28.

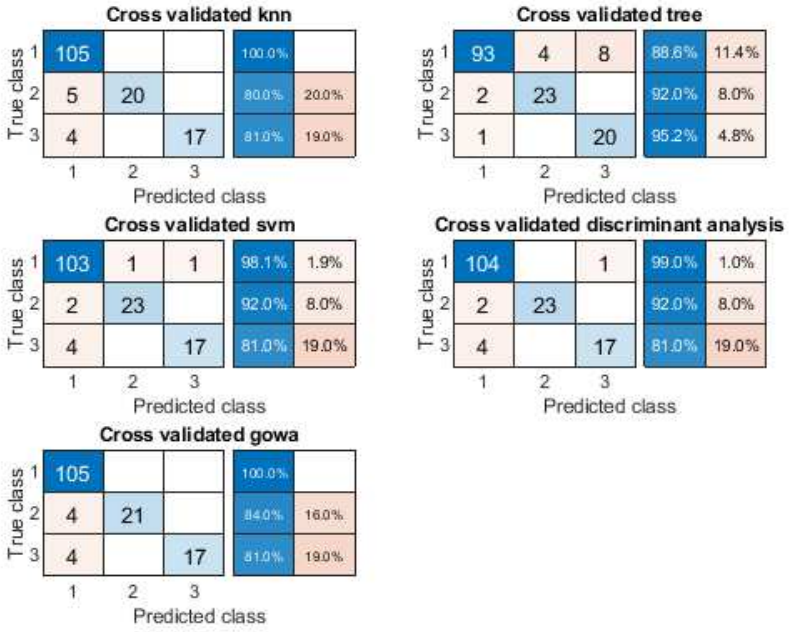

Figure 28. Confusion charts for cross-validated new-thyroid data.

As expected, all the classifiers performed quite well. The decision tree classifier has most false predictions, but in general all classifiers predicted most of the instances correctly. Confusion charts for the 30% test data for all the models are depicted in Figure 29.

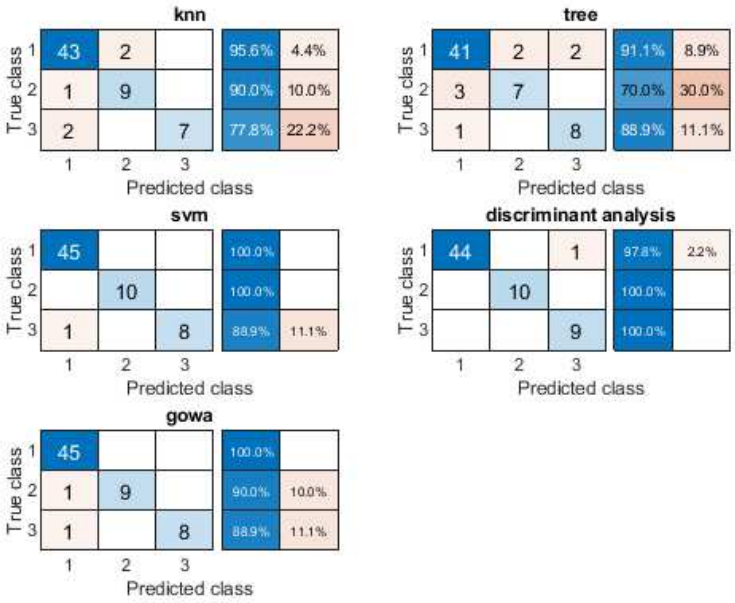

Figure 29. Confusion charts for New-thyroid test data.

There are no large variations in the confusion charts between different classifiers, but SVM, discriminant analysis and the GOWA-variant seem to perform a little bit better than others for this particular test data. All the performance measurement figures that were used to compare the classifiers are gathered in a table format in Figure 30. The discriminant analysis classifier has the best fitness function value.

	sn1	sn2	sn3	accuracy	cross_val_accuracy	fitness
knn	0.95556	0.9	0.77778	0.92188	0.9404	0.902
tree	0.91111	0.7	0.88889	0.875	0.90066	0.856
svm	1	1	0.88889	0.98438	0.94702	0.975
discriminant	0.97778	1	1	0.98438	0.95364	0.988
gowa	1	0.9	0.88889	0.96875	0.94702	0.95115

Figure 30. New-thyroid results.

But as said, there were variations between different runs with the different random data partitions. This is probably due to the small sample size. In Figure 31 there is a table, where fitness function values are visible for six different random data partitions (for 30/70 division). Also the mean values for these six different rounds are visible in the table.

	fit1	fit2	fit3	fit4	fit5	fit6	mean_fit	mean_accuracy
knn	0.882	0.912	0.877	0.914	0.878	0.902	0.894	0.9219
tree	0.882	0.865	0.771	0.893	0.855	0.856	0.854	0.8828
svm	0.904	0.938	0.901	0.939	0.939	0.975	0.933	0.9505
discriminant	0.976	0.939	0.924	0.939	0.951	0.988	0.953	0.9661
gowa	0.964	0.902	0.901	0.855	0.914	0.951	0.915	0.9427

Figure 31. New-thyroid accuracies with different random seed.

Since there are significant variations with majority of the classifiers between the six runs with different random seeds, it cannot be stated that some classifiers perform better than others. It can, however, be claimed that with a small data set it is hard to build a robust classification model.

4.5.3 Comparing classifiers with Default of credit card clients data

This data set differs significantly from the earlier two data sets due to a much larger number of observation and attributes. This makes the credit card default classification problem a bit more complex problem than the other two classification problems were. The third data set includes credit card client data with several predictors. Although, the data set is considerable larger than the first two data sets, a similar approach to the classification problem was used as what was used with the other two data sets. Only slight modification was done to the parameter optimization to ease the computational effort.

The credit card data set contains 30000 observations, and 23 parameters to be used as predictors. The target class labels are '0' and '1', where '0' refers to clients who have paid their monthly repayment of their credit card in time, and '1' refers to clients whose payments have defaulted. The target is to find out which clients will have defaults in their credit card payments, i.e., to which client credit should be given, and to which client it should not be given. First, data set was divided in two, as with earlier case, 40% was left out for testing and 60% was used for the model training and optimization. The following code was used for verifying that both training/optimization data and test data contain equal portion of both classes.

```
ClassData_count = height(ClassData)
cs = ClassData.class_label;
[classes,~,idx] = unique(cs);
nCount = accumarray(idx(:),1)
model_data_class0 = nCount(1)/ClassData_count
model_data_class1 = nCount(2)/ClassData_count

testData_count = height(testData)
cs = testData.class_label;
[classes,~,idx] = unique(cs);
nCount = accumarray(idx(:),1)
testdata_class0 = nCount(1)/testData_count
testdata_class1 = nCount(2)/testData_count
```

77.9% of the clients have taken care of their monthly payments duly and 22.1% of the clients had problems with their payments in both data sets (the model selection data set (60% of the original data) and the test data set (40% or the original data)).

The fitness function used for finding the optimum parameter and to finally rank the classifiers is similar to the fitness function that was used with Haberman's survival data set. The fitness function for credit card data set is as follows:

$$f_{fitness} = 0.35 * sn + 0.15 * sp + 0.5 * ACC$$

A slightly heavier weight is given for the sensitivity than for the specificity, since the assumption is, that it is more important to find out the clients whose payment will default than miss a few clients who would have been reliable. But similar to the other two data sets, again more domain knowledge would be required to be able to set the weights to better respond to the real-word requirements.

The hyperparameter optimization was used for finding the optimal distance measurement criterion and the optimal number of neighbors for the k-nearest neighbor classifier. The results of the hyperparameter optimization can be seen in Figure 32.

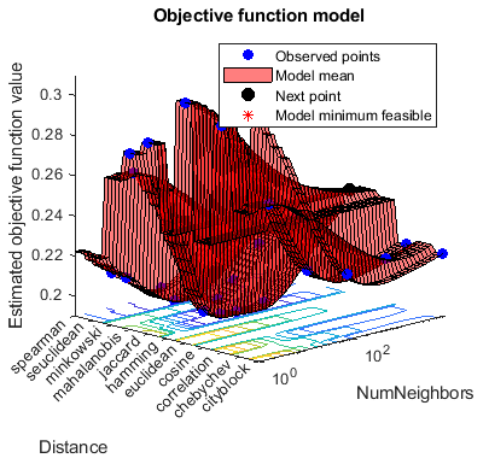

Figure 32. K-nearest neighbor parameter optimization for the credit data.

It was observed with all three data sets, that the hyperparameter optimization does not necessarily provide parameters, which would result the best classification accuracy and the best fitness function value. That is why the k-nearest neighbor classifier was also trained with a set of selected distance measurement criteria and with a selected range of neighbors in addition to the parameters provided by the hyperparameter optimization. The parameters, which resulted the best fitness function value, were selected for the final classification model. The hyperparameter optimization resulted ‘seuclidean’ for the best distance measurement criterion and 57 as the optimal number of neighbors. In addition to the hyperparameter optimization parameters, the knn-classifier was also trained with following distances {'cosine', 'euclidean', 'jaccard', 'chebychev', 'seuclidean'} and with [10:40] number of neighbors. The best fitness function value was achieved with ‘cosine’ distance and with 17 neighbors.

Support Vector Machine classifier requires heavy computations and thus, it does not suite well for complex classification problems. That is why the hyperparameter optimization was not run for the credit card data set. It was also observed with the other two data sets, that the hyperparameter optimization does not always provide the best possible parameters. So omitting the hyperparameter optimization, is unlikely to affect the final classification accuracy of the SVM classifier. Decision tree and discriminant analysis classifiers were trained the similar way as they were trained with the previous two data sets.

The GOWA-variant of the similarity classifier was first trained with different lambda values from within the range [0.1:0.3:5]. The best classification accuracy and also the best fitness function value was gained with lambda value 3.4., which can be observed from Figure 33.

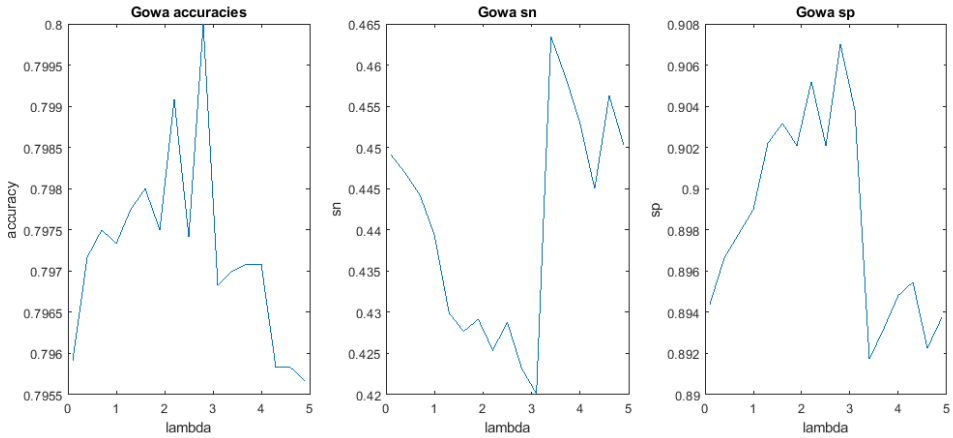

Figure 33. Different lambda values for credit card client data.

After the lambda value was selected, the GOWA-variant of the similarity classifier was trained with parameter values p and λ with both from within a range [0.25:0.25:5]. The mean classification accuracies are depicted in Figure 34. Similarly with the previous two data set, the maximum classification accuracy is in the middle of the grid, thus making the used parameter ranges feasible for the simulations.

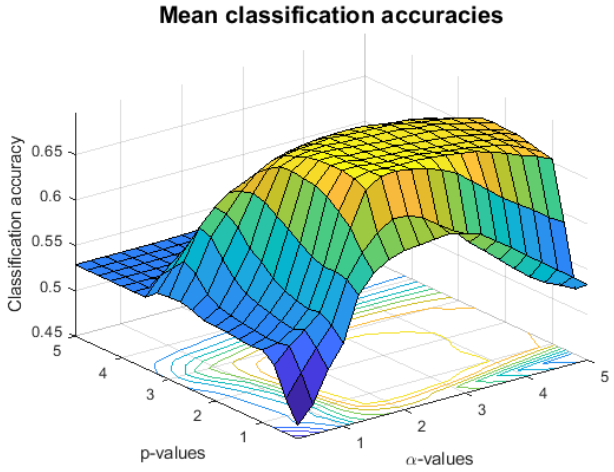

Figure 34. Mean classification accuracies for credit data.

The actual training and cross-validation was done the same way as it was done with the other two data sets. See the MATLAB codes from the appendix for details. The confusion charts for the cross-validation for all the five classifiers are depicted in Figure 35.

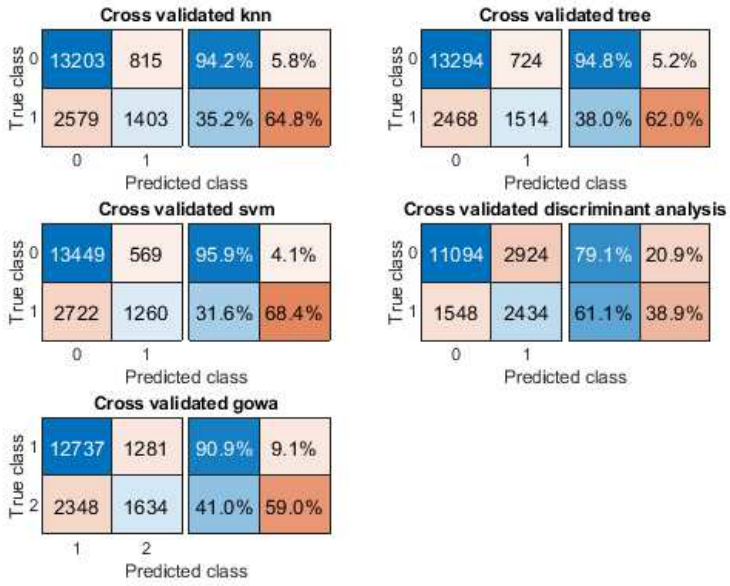

Figure 35. Confusion charts for cross-validated classifiers for credit data.

It can be seen from the confusion charts that most of classifiers were able to predict the target class '0' (clients who have made their payments duly) quite well. But predicting the clients whose payments would default, seem to be a hard task for the classifiers. Only one classifier, the discriminant analysis classifier, could predict more than half of the positive class ('1') members correctly. K-nearest neighbor, decision tree and SVM classifiers were only able to predict one third of the positive instances correctly. Confusion charts for the test data in Figure 36 give pretty much the same numbers as the confusion charts for the cross-validation data. Note that the confusion charts for the GOWA-variant of the similarity classifier has different labels for classes than the other classifiers. This is due to the fact that the GOWA-variant, which is used in this thesis, orders the classes and labels the classes in order starting from 1. Thus the class label '1' equals to the class label '0' and the class label '2' equals to the class label '1' in the GOWA-variant of the similarity classifier related confusion charts.

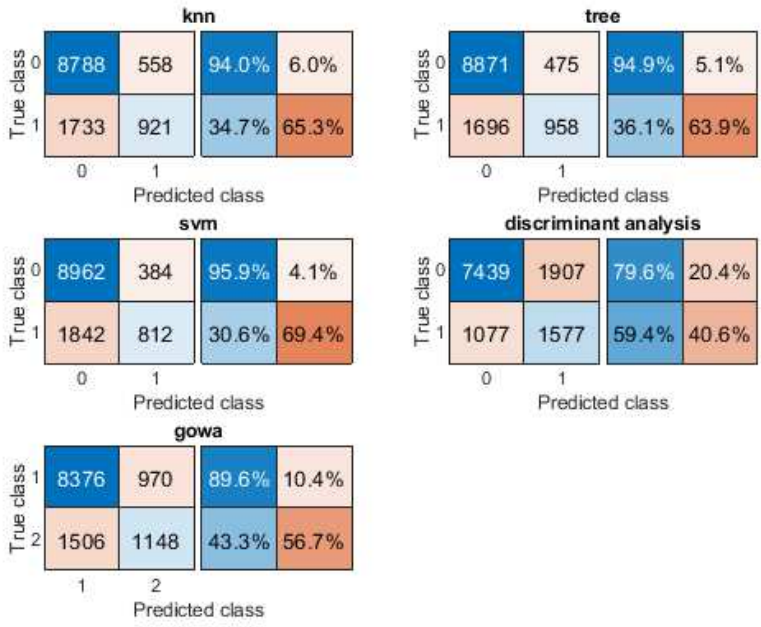

Figure 36. Confusion charts for credit card test data.

According to the confusion charts presented in Figure 35 and in Figure 36, it can be claimed that when the data set is large enough, cross-validation results and results gained with previously unseen test data are close to each other. More domain knowledge would be needed to be able to judge whether the generated classification models would be useful in a real business environment. Since although nearly half of the clients with problems in their payments were classifier incorrectly, it could still benefit the business if half of the possible defaulted cases would be pointed out beforehand and credit is then not granted to those clients.

The results of all of the five classifier are gathered in table format in Figure 37. From the table it can be seen, that there are no big differences between the classifiers. K-nearest neighbor, decision tree and SVM all provide quite similar results. It should be noted, that while discriminant analysis gives the best fitness function value and the best sensitivity value (detects best the defaulted clients) it makes most errors with clients who will pay their credit duly. So it should be evaluated which one is more critical to the company, to lose some good clients who would have made their payments duly or to be able to detect those clients who will end up with problems with their payments.

	sp	sn	accuracy	cross_val_accuracy	fitness
knn	0.9403	0.34702	0.80908	0.81144	0.66704
tree	0.94918	0.36096	0.81908	0.82267	0.67826
svm	0.95891	0.30595	0.8145	0.81717	0.65817
discriminant	0.79596	0.5942	0.75133	0.75156	0.70303
gowa	0.89621	0.43255	0.79367	0.79839	0.68266

Figure 37. Credit card client data results.

As this data set was significantly larger than the other two data sets, also the executing times play some role in selecting a suitable method for solving the classification problem. Training times with the computing power available for this thesis for all of the five classifiers are gathered in table format in Figure 38. The training time for the GOWA-variant of the similarity classifier is a training time with only one lambda value.

	training_time_in_minutes
knn	35.4
tree	2.5
svm	41.3
discriminant	0.3
gowa	5.7

Figure 38. Classifier training times for credit card data.

Training times in Figure 38 show that the classifier, which require more complex computation are much slower to train than other classifiers. Especially SVM gets slow when the observation count and the number of predictors increase. The training time for the GOWA-variant of the similarity classifier is directly proportional to the number of different parameter values used (p, α and λ). But if the parameter ranges are not too large, the training time for the GOWA-variant of the similarity classifier is feasible also for the larger data set.

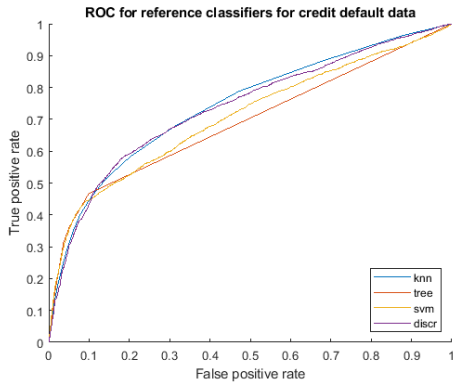

Figure 39. ROC curves for reference classifiers for credit card data.

The ROC curves for the four reference classifiers are depicted in Figure 39 for reference.

5 CONCLUSION

5.1 Answering the posed research questions

This thesis studies the GOWA-variant of the similarity classifier and investigates whether it is a useful method to be used in classification problems within the business context. This main research question was divided into three sub-question, which are used for supporting the answering to the main research question. The theoretical part of the thesis introduced the main research subject and defines the related terminology and methods that are needed to be able to understand the foundations of classification problems and more precisely the GOWA-variant of the similarity classifier. The practical part of the thesis contains benchmarking of a set of selected classical classifiers against the GOWA-variant of the similarity classifiers. Three different data sets are used for this benchmarking process. The benchmarking process is executed with mathematical tool called MATLAB.

The first sub-question posed is:

1. What previous academy research literature exists on similarity based classifiers and what are the results of the said previous research?

To answer this question, a literature review was performed and the results of that review are presented in chapter 3.4. The literature review shows that not a lot of research exists on similarity classifiers in the business context. Most of the research on similarity classifiers concentrates in the concept itself, further development of the method, and its capabilities and not so much on the possible application areas. It was observed the medical diagnostic data seems to be the most used type of data and thus medicine the most prolifically used application area for similarity classifiers. There were only a few research papers that concentrate directly on solving business related problem. Those papers were shortly introduced in Table 1. Credit scoring seems to be the most popular business related area for similarity classifiers. Some of the papers that were reviewed in chapter 3.4 relate to credit scoring data. One set of credit card client data is also used for this thesis in the classifier benchmarking process.

All in all, the previous research is “interested in” the quality of similarity classifiers, i.e., on how well they can perform and on which variants are the best. Most of the previous research can be categorized under mathematical or computer science research, but there is a research gap in the application of similarity classifiers to business related problems. There is a room for more research on how well similarity-based classifiers are suited for

solving business related problems and how they can be used to help business related decision making.

The second sub-question for this research is:

2. How do similarity-based classifiers work and especially how does the GOWA-variant of the similarity classifier work?

This sub-question is answered partly in the theoretical part of this thesis. Chapter 3.2 on similarity-based classification discusses the similarity-based classification in general and the chapter 3.3 on the GOWA-variant of the similarity classifier explains in more detail what the GOWA-variant of the similarity classifier is and how it is built. Classifiers in general aim to categorize new items to a set of pre-determined target classes. Similarity-based classifiers use a similarity measure to determine the correct class. There are different ways to determine the similarity measure and to aggregate similarities of individual attributes to a one common similarity value. The classifier used in this thesis uses generalized ordered weighted average operator in aggregation. When the GOWA-variant of the similarity classifier is trained ideal vectors that represent the target classes are generated first. The ideal vectors are representative vectors of centroid samples for each class. They include a set of measured features, also called predictors. A similarity value is calculated between a new sample and these ideal vectors (of all classes) and the sample is set to belong in the class from which the ideal vector it is most similar to the sample.

A MATLAB model of the GOWA-variant of the similarity based classifier is generated and tested with various data sets in chapter 4.5. The script that is used for executing the classification tasks in this thesis can be found from the appendix, see the codes under gowa_classifier.m. In that script both training and test data is first initialized so that all features are scaled between 0…1 and observations are ordered according to class (classes are labeled in a descending order, starting form 1). After that the ideal vectors are generated for training data for each fold. Then similarities are calculated between the ideal vectors and test data. The class label for each instance in test data is selected based on to the class with which the instance has the largest similarity value. After that the needed performance measure values, such as classification accuracy, are calculated.

The GOWA-variant of the similarity classifier was used to classify observations in three different data sets. Two of them were the same sets that were used in the research paper by Kurama et al. (2017). These two data sets (Haberman's survival data and New-thyroid data) were selected so that the results from this research can be compared to the research performed by Kurama et al. With Haberman's survival data Kurama et al. reached 76.3% classification accuracy with Basic RIM quantifier (the quantifier that is

used in this thesis). In this thesis the classification accuracy varied with different random data partition for training/cross-validation and test data, so it is hard to make a clear comparison. The whole training and testing procedure was repeated six times, for the last run 78.02% classification accuracy was achieved. The mean classification accuracy over the six consecutive runs was 73.8%. The figures are roughly in the same scale as what Kurama et. al received in their work. For the New-thyroid data set, the reference classification accuracy is 97.44% (taken from Kurama's et al. paper). The New-thyroid data set is quite small and it was observed in this thesis that classification accuracy varies a little when the random partition for initial model selection (the data used for cross-validation) and test data is done. Results in chapter 4.5.2 are reported with one random data split and also as a mean value of six different random splits. For one random split, the resulted accuracies are 94.7% for test data and 98.44% for the cross-validation. The mean value of the six subsequent runs is 95.5%. There are slight variations in the accuracy figures between different runs. But they are still quite close to the results provided by Kurama et al. The results between these two studies are not directly comparable as the data used for the classifier training and testing are not exactly the same. Even though the original data is the same data, it is split differently into training and test data sets, which will have an effect on the results – this is especially relevant since these two data sets are fairly small.

The third sub-question is:

3. How well does the studied similarity-based classifier(s) function in comparison with other selected classifiers?

To answer this question, a benchmarking process was performed were the GOWA-variant of the similarity classifier was compared against four classical classifiers. The different classifiers selected for the benchmarking are the K-nearest neighbor, discriminant analysis, decision tree and Support Vector Machine classifiers. All five classifiers were trained and tested with three different data sets. The exactly the same data was used for training and testing in each case to ensure comparability. The MATLAB scripts that were used for the benchmarking can be found in the appendix. The results for the comparison process are presented in detail in chapter 4.5. The classification accuracy is the most common way to assess classification performance, but the fitness function used in this thesis adds also to other measurement criteria to the performance evaluation, specifically the sensitivity and specificity, i.e., the classifiers capability to correctly classify positive and negative instances. The use of a fitness function in measuring performance of classifiers can be considered a small novel scientific contribution. Each measurement criteria was given a weight and the fitness function is built in the way that

output values are always from within the range 0…1 and for a perfect classifiers the result is 1.

With the Haberman's survival data, the GOWA-variant of the similarity classifier performed best. The GOWA-variant gave the value 0.685 for the fitness function (mean value of six runs), the second best decision tree classifier gained 0.673 fitness function value and results from the other tested classifiers were in the range 0.629…0.671. If we compare the mean classification accuracies, the discriminant analysis classifier has the best classification accuracy. See Figure 21 for details on the result.

New-thyroid data was the second benchmarking data set. It differs from the first one, as this is not a binary classification problem, instead there are three target classes. Therefore also the used fitness function is a bit different, than what is was for the Haberman's survival data. For New-thyroid data set, sensitivities for each target class are calculated and weighted equally in addition to classification accuracy to determine the fitness function value and thus the classifier performance. With this data set the results varied with different random data splits into cross-validation data and test data. Therefore it is harder to make a comparison between classifiers, different classifiers performed best in different runs. Only the decision tree classifier was clearly the worst performing classifier with this data in all runs. The results for a one run are gathered in Figure 30. To gain more reliable results, the classifiers were trained, cross-validated and tested six times with different random data split and the mean fitness function values of these six consecutive runs were calculated. If we look at those mean values, the GOWA-variant of the similarity classifier was third in performance - both in the mean fitness function value and in the mean classification accuracy. Discriminant analysis classifier performed best with this data set. But ranking the selected classifiers is not clear in this case, since the results varies quite a bit between different data splits. For instance the GOWA-variant of the similarity classifier outputs fitness function value 0.951 in the sixth run and 0.855 in the fourth run. The difference is quite significant. The performance of other classifiers also varies, but not that significantly. It can be claimed that it is hard to build a robust classifier with a rather small data set, but it might be even more so with the similarity classifier. At least this was the case with the New-thyroid data set.

The third data set is significantly larger than the other two sets and it is directly related to the business-related decision making. The third data set contains information of credit card clients and the target is to find out whether the clients are trustworthy or not, i.e., whether they will make their payments duly or will they be in default. As this data set was much more complex than the other two sets, also the running time plays some role when comparing different classifiers. But the main comparison was done with the fitness function as it was also done with the other two data sets. This time discriminant analysis classifier resulted the best fitness function value 0.703, but the GOWA-variant of the similarity classifier was only a slightly worse than that, giving a fitness function value

0.694. The rest of the classifiers had fitness function values from within the range 0.664…0.678. The notable difference with this data set was in the sensitivity values. The discriminant analysis classifier is clearly the best at recognizing true positive items, the GOWA-variant of the similarity classifier comes in second, but they are both worse than the other classifiers at recognizing true negative items. So it depends on what feature is valued the most, which classifier is most beneficial. For this thesis the fitness function was used to determine the ranking of classifiers. In the fitness function the sensitivity has a little more weight than the specificity. With this measurement criteria, discriminant analysis performs the best (fitness function value = 0.703) and the GOWA-variant of the similarity classifier is nearly as good as the discriminant analysis classifier (fitness function value = 0.694). The training times for each of the classifiers were also observed with this data set. If the lambda parameter value is determined beforehand for the GOWA-variant of the similarity classifier, then the training time is comparable with the other classifiers, but if we test a lot of different options for all parameters, then the training time gets fairly long. But it is the same with other classifiers as well, the more options you tests, the longer it takes. But as said, with one parameter set beforehand, training a similarity classifier is quite smooth even with bigger data.

The benchmarking process did not give a clear winner, as could be expected. Solving a classification problem solving is always a heuristic process, where you need to try many different methods and then choose the one that gives best results and is most suitable for the use case. There is no one method that would be suited best for all the classification problems. But as the GOWA-variant of the similarity classifier was compared against a set of classical classifiers, it was shown that its performance is in line with the other classifiers. It outperformed the other classifiers with one data, and did fairly well also with the other two data. After given and answer to all of the sub-questions, we can also answer to the main research question, which was whether the GOWA-variant of the similarity classifier is a useful classifier to be used in the business context. According to the findings in this research, and specifically to answer the main research question posed, the GOWA-variant of the similarity classifier seems to be a viable option when classifiers are selected to be used in the business context.

5.2 Criticism and future research directions

Both of the small data sets used in this study indicated that the original split into a training data and test data could affect significantly to the final classification accuracy. This study used a method where a predetermined amount of data (30% or 40% in this study) was left out as test data and the rest of the data was used for model training and cross-validation.

Another option could have been to divide the whole data set into training and test data, and repeat that random split several times. The final classification accuracy would then be calculated as a mean value of those repeated runs. For a smaller data sets, the repeated splits and mean value calculation could generate more robust results, but separating a part of data from a beginning for testing helps to avoid overfitting. For the smaller data sets, this thesis combined these two approaches. A part of the data was left out for testing at the beginning to avoid overfitting, but the whole procedure was repeated several times and mean values were calculated for the these consecutive runs. Since there are different options on how data is divided into training and test data, the comparison of different methods should always be done with exactly the same data. If random data split is used, the random seed should be predetermined to ensure repeatable results. This study confirms the two aforesaid requirements.

As said several times already earlier in this study, choosing a classifier is always a heuristic process, and there is no one method that is best suited for all classification problems. Thus, the results gained in this study cannot not be generalized to a wider content, i.e., it cannot be guaranteed that classifiers, which performed best with the data sets used in this thesis, will also perform best with some other data sets.

This research is quite narrow with only three data sets and only one of them being a business related data, it still did not show any reason why similarity based classification should not be used in business related decision making. It was observed that for small data sets the results varied a lot depending on the used test data. So in the future more benchmarking should be performed with larger data sets to gain more insights of the performance of the GOWA-variant of the similarity classifier. The data sets that were used in this research contained only numerical data and there were no missing items. Further research should be performed to find out if the GOWA-variant of the similarity classifier is suitable for data sets were there are also categorical attributes. Further research could also study how well similarity classifier deal with missing items in predictors. This study only used data from the UCI Machine Learning Repository, which is a common source for data in machine learning research. In the future it would be interesting to compare different classifiers with some real world business data, were input data is probably not that uniform and more pre-processing of the data is needed before training the classifiers can be performed.

REFERENCES

Allaby, M. (2013). *A dictionary of geology and earth sciences* Oxford University Press.

Berthold, M. R., Borgelt, C., Höppner, F., & Klawonn, F. (2010). *Guide to intelligent data analysis: How to intelligently make sense of real data* Springer Science & Business Media.

Bonabeau, E. (2003). Don't trust your gut. *Harvard Business Review,* 81(5), 116-23, 130.

Breastcancer.org. (2018), Lymph node involvement. <https://www.breastcancer.org/symptoms/diagnosis/lymph_nodes >, retrieved 6.3.2019.

Cazzanti, L. (2007). Generative models for similarity-based classification. (Dissertation, University of Washington).

Cazzanti, L. – Gupta, M. R. – Srivastava, S. (2009). Fusing similarities and euclidean features with generative classifiers. *Information Fusion, 2009. FUSION'09. 12th International Conference On,* pp. 224-231.

Cengiz, Ö. (2017), Neural networks and pattern recognition using MATLAB -k_n – nearest neighbor estimation. <https://www.byclb.com/TR/Tutorials/neural_networks/ >, retrieved 18.10.2018.

Chen, Y. – Garcia, E. K. – Gupta, M. R. – Rahimi, A. – Cazzanti, L. (2009). Similarity-based classification: Concepts and algorithms. *Journal of Machine Learning Research,* 10(Mar), 747-776.

Cortes, C. – Vapnik, V. (1995). Support-vector networks. *Machine Learning,* 20(3), 273-297.

Davenport, T. H., & Harris, J. G. (2007). *Competing on analytics: The new science of winning* Harvard Business Press.

Detyniecki, M. (2001). Fundamentals on aggregation operators. *This Manuscript is Based on Detyniecki's Doctoral Thesis and can be Downloaded From,*

DeZyre. (2018), Top 10 machine learning algorithms. <https://www.dezyre.com/article/top-10-machine-learning-algorithms/202 >, retrieved 17.10.2018.

Eriksson, P., & Kovalainen, A. (2015). *Qualitative methods in business research: A practical guide to social research* Sage.

Fink, A. (2005). *Conducting research literature reviews: From the internet to paper* Sage.

Forbes technology council. (2018), 15 business applications for artificial intelligence and machine learning. <https://www.forbes.com/sites/gordon-kelly/2018/10/02/apple-iphone-xs-max-charging-battery-life-upgrade-price-cost-iphone-xr/#65f162b81a07 >, retrieved 3.10.2018.

Fullér, R. – Majlender, P. (2003). On obtaining minimal variability OWA operator weights. *Fuzzy Sets and Systems,* 136(2), 203-215.

Garg, R. (2018), 7 types of classification algorithms. <https://www.analyticsindiamag.com/7-types-classification-algorithms/ >, retrieved 17.10.2018.

Gonçalves, L. – Subtil, A. – Oliveira, M. R. – de Zea Bermudez, P. (2014). ROC curve estimation: An overview. *REVSTAT-Statistical Journal,* 12(1), 1-20.

Hevner, A. R. – March, S. T. – Park, J. – Ram, S. (2004). Design science in information systems research. *MIS Quarterly,* 28(1), 75-105.

Hossin, M. – Sulaiman, M. (2015). A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process,* 5(2), 1.

Imran, M. – Afzal, M. T. – Qadir, M. A. (2016). Malware classification using dynamic features and hidden markov model. *Journal of Intelligent & Fuzzy Systems,* 31(2), 837-847.

Janssen, D. (2018), Business analytics (BA). <https://www.techopedia.com/definition/344/business-analytics-ba >, retrieved 13.2.2018.

Janusz, A. (2008). Similarity relation in classification problems. *Rough Sets and Current Trends in Computing, Proceedings,* 5306, 211-222.

Kotsiantis, S. (2007). Supervised machine learning: A review of classification techniques. *Informatica,* 31(3), 249-268.

Kuechler, B. – Vaishnavi, V. (2004). Design science research in information systems. *URI: Http://Www.Desrist.Org/Design-Research-in-Information-Systems/.Online,*

Kurama, O. – Luukka, P. – Collan, M. (2017). A similarity classifier with generalized ordered weighted averaging operator. *Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS), 2017 Joint 17th World Congress of International,* pp. 1-6.

Kurama, O. – Luukka, P. – Collan, M. (2015). Credit analysis using a combination of fuzzy robust PCA and a classification algorithm. *Scientific Methods for the Treatment of Uncertainty in Social Sciences,* 377, 19-29.

Kurama, O. – Luukka, P. – Collan, M. (2016a). AN N-ARY lambda-AVERAGING BASED SIMILARITY CLASSIFIER. *International Journal of Applied Mathematics and Computer Science,* 26(2), 407-421.

Kurama, O. – Luukka, P. – Collan, M. (2016b). A similarity classifier with bonferroni mean operators. *Advances in Fuzzy Systems,* , 7173054.

Le, J. (2018), A tour of the top 10 algorithms for machine Learning Newbies. <https://towardsdatascience.com/a-tour-of-the-top-10-algorithms-for-machine-learning-newbies-dde4edffae11 >, retrieved 17.10.2018.

Lichman, M. (2013), UCI machine learning repository. <http://archive.ics.uci.edu/ml >, retrieved 2018.

Liu, X. – Han, S. (2008). Orness and parameterized RIM quantifier aggregation with OWA operators: A summary. *International Journal of Approximate Reasoning,* 48(1), 77-97.

Luukka, P. (2010a). Nonlinear fuzzy robust PCA algorithms and similarity classifier in bankruptcy analysis. *Expert Systems with Applications,* 37(12), 8296-8302.

Luukka, P. (2010b). Nonlinear fuzzy robust PCA algorithms and similarity classifier in bankruptcy analysis. *Expert Systems with Applications,* 37(12), 8296-8302.

Marr, B. (2016), The top 10 AI and machine learning use cases everyone should know about. <https://www.forbes.com/sites/bernard-marr/2016/09/30/what-are-the-top-10-use-cases-for-machine-learning-and-ai/#373f66b094c9 >, retrieved 3.10.2018.

Mathworks. (2018), What is machine learning?
3 things you need to know. <https://www.mathworks.com/discovery/machine-learning.html >, retrieved 15.10.2018.

Meyer, D. – Wien, F. T. (2001). Support vector machines. *R News,* 1(3), 23-26.

Mitroff, I. I. – Betz, F. – Pondy, L. R. – Sagasti, F. (1974). On managing science in the systems age: Two schemas for the study of science as a whole systems phenomenon. *Interfaces,* 4(3), 46-58.

Provost, F., & Fawcett, T. (2013). *Data science for business: What you need to know about data mining and data-analytic thinking* " O'Reilly Media, Inc.".

Qiu, J. – Wu, Q. – Ding, G. – Xu, Y. – Feng, S. (2016). A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing,* 2016(1), 67.

Raschkam, S. (2014), Linear discriminant analysis. <https://sebastian-raschka.com/Articles/2014_python_lda.html >, retrieved 19.2.2019.

Rouse, M. (2017), Business analytics (BA). <https://searchbusinessanalytics.techtarget.com/definition/business-analytics-BA >, retrieved 25.9.2018.

Rouse, M. (2018), Machine learning (ML). <https://searchenterpriseai.techtarget.com/definition/machine-learning-ML >, retrieved 3.10.2018.

Samuel, A. L. (1969). Some studies in machine learning using the game of checkers. II—Recent progress. *Annual Review in Automatic Programming,* 6, 1-36.

Saunders, M., Lewis, P., & Thornhill, A. (2009). *Research methods for business students* (5th ed.) Pearson Education Limited.

Shirkhorshidi, A. S. – Aghabozorgi, S. – Wah, T. Y. (2015). A comparison study on similarity and dissimilarity measures in clustering continuous data. *PloS One,* 10(12)

Shmueli, G., Patel, N. R., & Bruce, P. C. (2016). *Data mining for business analytics: Concepts, techniques, and applications with XLMiner* John Wiley & Sons.

Siegel, E. (2013). *Predictive analytics: The power to predict who will click, buy, lie, or die* John Wiley & Sons.

Skabar, A. (2013). Direction-of-change financial time series forecasting using a similarity-based classification model. *Journal of Forecasting,* 32(5), 409-422.

Sokolova, M. – Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management,* 45(4), 427-437.

Spackman, K. A. (1989). Signal detection theory: Valuable tools for evaluating inductive learning. *Proceedings of the Sixth International Workshop on Machine Learning,* pp. 160-163.

Steffens, T. (2005). In Kaelbling L. S.,A. (Ed.), *Partial and vague knowledge for similarity measures*

Takeda, H. – Veerkamp, P. – Yoshikawa, H. (1990). Modeling design process. *AI Magazine,* 11(4), 37.

Tan, P., Steinbach, M., & Kumar, V. (2006). Classification: Basic concepts, decision trees, and model evaluation. *Introduction to data mining* (pp. 145-195) Pearson Addison Wesley.

Tao, A. (2018), **The most successful companies today are data-driven**. <http://blogs.mccombs.utexas.edu/mstc/2018/04/16/the-most-successful-companies-today-are-data-driven/ >, retrieved 19.3.2019.

Thoma, M. (2018), Evaluation of binary classifiers. <https://martin-thoma.com/binary-classifier-evaluation/ >, retrieved 18.2.2019.

Torra, V., & Narukawa, Y. (2007). *Modeling decisions: Information fusion and aggregation operators* Springer Science & Business Media.

Tversky, A. (1977). Features of similarity. *Psychological Review,* 84(4), 327.

Varone, M. – Mayer, D. – Melegari, A. (2018), What is machine learning? A definition. <https://www.expertsystem.com/machine-learning-definition/ >, retrieved 3.10.2018.

Wang, S. – Wang, H. (2010). Towards innovative design research in information systems. *Journal of Computer Information Systems,* 51(1), 11-18.

Wang, Z., Chong, C. S., Lan, L., Yang, Y., Ho, S. B., & Tong, J. C. (2016). *Fine-grained sentiment analysis of social media with emotion sensing*

Watson, Jane Webster Richard T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS Quarterly,* 26, 2.

Yager, R. R. (1988). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man, and Cybernetics,* 18(1), 183-190.

Yager, R. R. (1996). Quantifier guided aggregation using OWA operators. *International Journal of Intelligent Systems,* 11(1), 49-73.

Yager, R. R. (2004). Generalized OWA aggregation operators. *Fuzzy Optimization and Decision Making,* 3(1), 93-107.

Zhou, L. – Chen, H. (2014). Generalized ordered weighted proportional averaging operator and its application to group decision making. *Informatica,* 25(2), 327-360.

APPENDIX

knn_classifier.m:

```
%% Train K-nearest neighbor classifier
rng(1);
Mdl = fitcknn(predictors, ClassData.class_label, 'OptimizeHyperparame-
ters', ...
      'auto', 'HyperparameterOptimizationOptions', ...
      struct('AcquisitionFunctionName', 'expected-improvement-plus'));
opdist = Mdl.ModelParameters.Distance;
opneigbors = Mdl.ModelParameters.NumNeighbors;

fitCum = zeros(length(distances),length(numNeighbors));
for j=1:length(distances)
   for i=1:length(numNeighbors)
       knnmdl = fitcknn(predictors, ClassData.class_label, ...
            'Distance', string(distances(j)), ...
            'NumNeighbors', numNeighbors(i), ...
            'DistanceWeight', 'Equal', 'Standardize', true);
       cvknnmdl = crossval(knnmdl, 'CVPartition', cv_data);
       [predictions, score] = kfoldPredict(cvknnmdl);
       [Accuracy, sn, sp] = evaluate_classifier(predictions,
                 ClassData.class_label, classes, 0, '');
       if (length(classes)==2)
          fitness = calc_fit(Accuracy, sn, sp, fitness_weights);
       else
          fitness = calc_fit_multiclass(Accuracy, sn(1),sn(2),sn(3),…
                    fitness_weights);
       end
       fitCum(j,i) = fitness;
   end
end

% train model with results of hyperparameter optimization
knnmdl = fitcknn(predictors, ClassData.class_label, 'Distance',…
         opdist, ...
                 'NumNeighbors', opneigbors, ...
                 'DistanceWeight', 'Equal', 'Standardize', true);
cvknnmdl = crossval(knnmdl, 'CVPartition', cv_data);
[predictions, score] = kfoldPredict(cvknnmdl);
[Accuracy, sn, sp] = evaluate_classifier(predictions,…
                     ClassData.class_label, classes, 0, '');
if (length(classes)==2)
   optfitness = calc_fit(Accuracy, sn, sp, fitness_weights);
else
   optfitness = calc_fit_multiclass(Accuracy, sn(1),sn(2),sn(3), …
                fitness_weights);
end
%% find maximum fitness function value
maxfit = max(max(fitCum));
[j1,i1]=find(maxfit==fitCum);

% train the model with optimal parameters
if (maxfit>optfitness)
   knnmdl = fitcknn(predictors, ClassData.class_label, ...
            'Distance', string(distances(j1(1))), ...
            'NumNeighbors', numNeighbors(i1(1)), ...
            'DistanceWeight', 'Equal', 'Standardize', true);
   cvknnmdl = crossval(knnmdl, 'CVPartition', cv_data);
   optimal_knn_distance = string(distances(j1(1)))
```

```
   optimal_num_neighbors = numNeighbors(i1(1))
end

cvAccuracy = (1 - kfoldLoss(cvknnmdl, 'LossFun', 'ClassifError'));
```

tree_classifier.m:

```
function [treemdl, cvtreemdl, treecvAccuracy] = …
          tree_classifier(ClassData, cv_data, predictors, …
                          classes, fitness_weights)

%% Train Decision tree classifier
i_val = [];
k_val = [];
Fit_Cum = [];
for i=1:10
  for k=1:10
    treemdl = fitctree(predictors, ClassData.class_label,…
              'SplitCriterion', 'gdi', ...
              'MaxNumSplits', i*2+2, 'MinLeafSize', 10+k*2, …
              'Surrogate', 'off');
    cvtreemdl = crossval(treemdl, 'CVPartition', cv_data);
    [predictions, score] = kfoldPredict(cvtreemdl);
    [Accuracy, sn, sp] = evaluate_classifier(predictions,…
                         ClassData.class_label, classes, 0, '');
    if (length(classes)==2)
        fitness = calc_fit(Accuracy, sn, sp, fitness_weights);
    else
        fitness = calc_fit_multiclass(Accuracy, sn(1),sn(2),sn(3),…
                  fitness_weights);
    end
    Fit_Cum = [Fit_Cum fitness];
    i_val = [i_val i];
    k_val = [k_val k];
  end
end
[M, I] = max(Fit_Cum);

treemdl = fitctree(predictors, ClassData.class_label,…
                   'SplitCriterion', 'gdi', ...
                   'MaxNumSplits', i_val(I)*2+2, 'MinLeafSize',…
                   10+k_val(I)*2, 'Surrogate', 'off');
cvtreemdl = crossval(treemdl, 'CVPartition', cv_data);
treecvAccuracy = (1 - kfoldLoss(cvtreemdl, 'LossFun',…
                  'ClassifError'));
```

svm_classifier.m:

```
function [svmmdl,cvsvmmdl,svmcvAccuracy] = svm_classifier(ClassData, …
          cv_data, predictors, classes, fitness_weights, doHyperOpt)

%% Train SVM classifier
if (doHyperOpt==1)
   rng(1);
   opts =  struct('Optimizer','bayesopt','ShowPlots',true,…
           'CVPartition',cv_data,...
           'AcquisitionFunctionName','expected-improvement-plus');
   rng(1);
   svmmod = fitcsvm(predictors,ClassData.class_label,…
            'KernelFunction','rbf',...
            'OptimizeHyperparameters','all',…
            'HyperparameterOptimizationOptions',opts);
```

```
   scale_f = [1e-5 1e-4 1e-3 1e-2 1e-1 1 10 100 1000 10000 100000];
else
   scale_f = [1e-4 1e-2 1 10 100];
end

%Retrain the SVM classifier, but adjust the 'KernelScale' and 'BoxCon-
straint' name-value pair arguments.
svmmod2 = fitcsvm(predictors, ClassData.class_label, ...
                  'KernelFunction', 'gaussian', ...
                  'PolynomialOrder', [], 'KernelScale', 'auto', ...
                  'BoxConstraint', 1, ...
                  'Standardize', true);
scale = svmmod2.KernelParameters.Scale;

Fit_Cum = [];
for i=1:(length(scale_f))
    svmmdl = fitcsvm(predictors, ClassData.class_label, ...
                     'KernelFunction', 'gaussian', ...
                     'PolynomialOrder', [], 'KernelScale',…
                      scale*scale_f(i), ...
                     'BoxConstraint', scale_f(i), ...
                     'Standardize', true);
    cvsvmmdl = crossval(svmmdl, 'CVPartition', cv_data);
    [predictions, score] = kfoldPredict(cvsvmmdl);
    [Accuracy, sn, sp] = evaluate_classifier(predictions,…
                        ClassData.class_label, classes, 0, '');
    fitness = calc_fit(Accuracy, sn, sp, fitness_weights);
    Fit_Cum = [Fit_Cum fitness];
end

if (doHyperOpt==1)
    if (isnan(svmmod.HyperparameterOptimizationResults.XAtMinObjec-
tive.KernelScale))
       scale = 1;
    else
       scale = svmmod.HyperparameterOptimizationResults.XAtMinObjec-
tive.KernelScale;
    end
    svmmdl = fitcsvm(predictors, ClassData.class_label, ...
                     'KernelFunction', string(svmmod.HyperparameterOp-
timizationResults.XAtMinObjective.KernelFunction), ...
                     'BoxConstraint',…
svmmod.HyperparameterOptimizationResults.XAtMinObjective.BoxCon-
straint,...
                     'KernelScale', scale,...
                     'Standardize', true);
    cvsvmmdl = crossval(svmmdl, 'CVPartition', cv_data);
    [predictions, score] = kfoldPredict(cvsvmmdl);
    [Accuracy, sn, sp] = evaluate_classifier(predictions,…
                       ClassData.class_label, classes, 0, '');
    fitness = calc_fit(Accuracy, sn, sp, fitness_weights);
    Fit_Cum = [Fit_Cum fitness];
end

[M, I] = max(Fit_Cum);

if (I==length(scale_f)+1)
   kScale    = svmmod.HyperparameterOptimizationResults.XAtMinObjec-
tive.KernelScale
   BoxC      = svmmod.HyperparameterOptimizationResults.XAtMinObjec-
tive.BoxConstraint
```

```
   KFunction =  string(svmmod.HyperparameterOptimizationRe-
sults.XAtMinObjective.KernelFunction)
else
   kScale = scale*scale_f(I)
   BoxC   = scale_f(I)
   KFunction =  'gaussian'
end

svmmdl = fitcsvm(predictors, ClassData.class_label, ...
                 'KernelFunction', KFunction, ...
                 'PolynomialOrder', [], 'KernelScale', kScale, ...
                 'BoxConstraint', BoxC, ...
                 'Standardize', true);
cvsvmmdl = crossval(svmmdl, 'CVPartition', cv_data);

svmcvAccuracy = (1 - kfoldLoss(cvsvmmdl, 'LossFun', 'ClassifError'));
```

discriminant_classifier.m:

```
function [dismdl, cvdismdl, cvdisAccuracy] = discriminant_classi-
fier(ClassData, cv_data, predictors, classes, fitness_weights, discri-
minant_types)

%% Train Discriminant Analysis classifier
gammap = [0, 1];
fillp = {'off', 'on'};
i_val = [];
k_val = [];
n_val = [];
Fit_Cum = [];
for i=1:length(discriminant_types)
    for k=1:2
       for n=1:2
         dismdl = fitcdiscr(predictors, ClassData.class_label, ...
                   'DiscrimType', string(discriminant_types(i)), ...
                   'FillCoeffs', string(fillp(n)), ...
                   'Gamma', gamma(k));
         i_val = [i_val i];
         k_val = [k_val k];
         n_val = [n_val n];
         cvdismdl = crossval(dismdl, 'CVPartition', cv_data);
         [predictions, score] = kfoldPredict(cvdismdl);
         [Accuracy, sn, sp] = evaluate_classifier(predictions,…
                    ClassData.class_label, classes, 0, '');
         if (length(classes)==2)
            fitness = calc_fit(Accuracy, sn, sp, fitness_weights);
         else
            fitness = calc_fit(Accuracy, sn(1), sp, fitness_weights);
         end
         Fit_Cum = [Fit_Cum fitness];
       end
    end
end

[M, I] = max(Fit_Cum);

dismdl = fitcdiscr(predictors, ClassData.class_label, ...
                   'DiscrimType', …
                   string(discriminant_types(i_val(I))), ...
                   'FillCoeffs', string(fillp(n_val(I))), ...
                   'Gamma', gamma(k_val(I)));
cvdismdl = crossval(dismdl, 'CVPartition', cv_data);
```

```
cvdisAccuracy = (1 - kfoldLoss(cvdismdl, 'LossFun', 'ClassifError'));
```

gowa_classifier.m:

```
function [Accuracy, class, sn, sp, truelabels, cvfitness, cvclass,…
truecvlabels, fitness_f] = gowa_classifier(ClassData, cv, testData,…
classes, fitness_weights, v, c, p, m, lambda, pl)

measure = 1; % Which quantifier in OWA operators you want to use.

fitness = zeros(1,cv.NumTestSets);
fit = zeros(1,cv.NumTestSets);
meanfit = zeros(length(p),length(m));
Vars = zeros(length(p),length(m));
Maxsf = zeros(length(p),length(m));
Minsf = zeros(length(p),length(m));

for j = 1:length(m)
    for i = 1:length(p)
        y = [p(i), m(j), measure, lambda];
        for k = 1:cv.NumTestSets
            trIdx = cv.training(k);
            teIdx = cv.test(k);
            datalearn = table2array(ClassData(trIdx, :));
            datatest  = table2array(ClassData(teIdx, :));
            [datatest, tc_sizes, cstart_idx] = …
                                init_data(datatest,v,c);
            [datalearn, lc_sizes, cstart_idx] = …
                                init_data(datalearn,v,c);
            ideal_vec = idealvectors(datalearn, y);
            [fitness(k), class, Simil] = calcfit(datatest, …
                                        ideal_vec, y);
            [acctmp, sn, sp] = evaluate_classifier(class,…
                               datatest(:,end), classes, 0, '');
            if (length(classes)==2)
                fit(k) = calc_fit(fitness(k),sn,sp,fitness_weights);
            else
                fit(k) = calc_fit_multiclass(fitness(k),sn(1),sn(3),…
                         sn(3), fitness_weights);
            end
        end
        meanfit(i,j) = mean(fit);
        Vars(i,j)  = var(fit);
        Maxsf(i,j) = max(fit);
        Minsf(i,j) = min(fit);
    end
    fitness=[];
    fit = [];
end
maxfit = max(max(meanfit));
[p1,m1]=find(maxfit==meanfit);
% if maximum value is there more than once, take the first one and
find out the actual parameter
% p and m values for that index
p_o = p(p1(1));
m_o = m(m1(1));

data_h = table2array(ClassData);
data_h = init_data(data_h, v, c);
testArraytmp = table2array(testData);
testArray = init_data(testArraytmp, v, c);
```

```
truelabels = testArray(:,end);
truecvlabels = data_h(:,end);

% find optimum parameter for max fitness function and calculate test-
data values with those parameters
y = [p_o, m_o, measure, lambda];
% calculate ideal vector with the best parameters for the whole train-
ing data
final_ideal_vec = idealvectors(data_h, y);
% calculate cross-validation fitness
[cvfitness, cvclass, cvSimil] = calcfit(data_h, final_ideal_vec, y);
% calculate fitness for the test data (that data has not been used for
training)
[fitness, class, Simil] = calcfit(testArray, final_ideal_vec, y);
% calculate sensitivity & specificity for the final classifier with
test data
[acc, sn, sp] = evaluate_classifier(class, testArray(:,end), …
                classes, 0, '');
% calculate fitness function value for test data
if (length(classes)==2)
   fitness_f = calc_fit(fitness, sn, sp, fitness_weights);
else
   fitness_f = calc_fit_multiclass(fitness, sn(1), sn(2), sn(3), fit-
ness_weights);
end

Accuracy = fitness;

if (pl==1)
    [X,Y] = meshgrid(m,p);
    figure
    surfc(X,Y,meanfit)
    title('Mean classification accuracies','FontSize',15)
    xlabel('\alpha-values')
    ylabel('p-values')
    zlabel('Classification accuracy')
    end
    clear Y
end
```

multisvm_classifier.m:

```
function [svmmdl,cvsvmmdl,svmcvAccuracy] = …
         multisvm_classifier(ClassData, cv_data, predictors, …
                             classes, fitness_weights)

%% Train SVM classifier
opts = struct('Optimizer','bayesopt', 'ShowPlots',true, …
       'CVPartition',cv_data,...
       'AcquisitionFunctionName','expected-improvement-plus');
rng(1);
svmmod = fitcecoc(predictors,ClassData.class_label,…
         'OptimizeHyperparameters', 'auto',…
         'HyperparameterOptimizationOptions', opts);

%Retrain the SVM classifier, but adjust the 'KernelScale' and 'BoxCon-
straint' name-value pair arguments.
template = templateSVM('KernelFunction', 'gaussian', ...
                       'PolynomialOrder', [], ...
                       'KernelScale', 'auto' , ...
                       'BoxConstraint', 1, ...
```

```
                        'Standardize', true);
classificationSVM2 = fitcecoc(predictors, ClassData.class_label, ...
                    'Learners', template);

% hyperpamater optimization is quite slow for this classifier. That is
% why it is not run each time, the values should be the same since %
rng(1) was called before the above optimal values were generated
opt_coding =…
 svmmod.HyperparameterOptimizationResults.XAtMinObjective.Coding;
opt_scale = …
svmmod.HyperparameterOptimizationResults.XAtMinObjective.KernelScale;
opt_box = …
svmmod.HyperparameterOptimizationResults.XAtMinObjective.BoxCon-
straint;

k_func = {'gaussian', 'linear', 'polynomial'};
Fit_Cum = [];
for i=1:4
   if (i==4)
       tmpl = templateSVM('KernelFunction', 'gaussian', ...
                        'PolynomialOrder', [], ...
                        'KernelScale', opt_scale, ...
                        'BoxConstraint', opt_box, ...
                        'Standardize', true);
       svmmdl = fitcecoc(predictors, ClassData.class_label, ...
                    'Learners', tmpl, ...
                    'Coding', string(opt_coding));
   else
       tmpl = templateSVM('KernelFunction', string(k_func(i)), ...
                        'PolynomialOrder', [], ...
                        'KernelScale', 'auto', ...
                        'BoxConstraint', 1, ...
                        'Standardize', true);
       svmmdl = fitcecoc(predictors, ClassData.class_label, ...
                     'Learners', tmpl);
   end
   cvsvmmdl = crossval(svmmdl, 'CVPartition', cv_data);
   [predictions, score] = kfoldPredict(cvsvmmdl);
   [Accuracy, sn, sp] = evaluate_classifier(predictions,
                    ClassData.class_label, classes, 0, '');
   fitness = calc_fit_multiclass(Accuracy, sn(1), sn(2), sn(3),…
                                 fitness_weights);
   Fit_Cum = [Fit_Cum fitness];
end

[M, I] = max(Fit_Cum);

if (I==4)
   kernel_func = 'gaussian';
else
   kernel_func = k_func(I);
end

tmpl = templateSVM('KernelFunction', string(kernel_func), ...
                   'PolynomialOrder', [], ...
                   'KernelScale', 'auto', ...
                   'BoxConstraint', 1, ...
                   'Standardize', true);
svmmdl = fitcecoc(predictors, ClassData.class_label, ...
                   'Learners', tmpl);
```

```
cvsvmmdl = crossval(svmmdl, 'CVPartition', cv_data);

svmcvAccuracy = (1 - kfoldLoss(cvsvmmdl, 'LossFun', 'ClassifError'));
```

calc_fit.m:

```
function fitness = calc_fit(Accuracy, sn, sp, weights)

fitness = weights(1)*Accuracy + weights(2)*sn + weights(3)*sp;
```

calc_fit_multiclass.m:

```
unction fitness = calc_fit_multiclass(Accuracy, sn1, sn2, sn3,…
                                      weights)

fitness = weights(1)*Accuracy + weights(2)*sn1 + weights(3)*sn2…
          + weights(4)*sn3;
```

evaluate_classifier.m:

```
function [Accuracy, sn, sp] = evaluate_classifier(predictions,…
             truelabels, classes, plot_ch, charttitle)

[cm,cl] = confusionmat(truelabels, predictions);

if (plot_ch==1)
   figure
   confusionchart(truelabels,predictions, 'ColumnSummary',…
                  'column-normalized', ...
                 'RowSummary','row-normalized');
   title(charttitle);
end

if (length(classes) == 2)
   FP = cm(cl==classes(1), cl==classes(2));
   FN = cm(cl==classes(2), cl==classes(1));
   TP = cm(cl==classes(2), cl==classes(2));
   TN = cm(cl==classes(1), cl==classes(1));
   sn = TP / (TP+FN);
   sp = TN / (TN+FP);
   Accuracy = (TP+TN)/(TP+TN+FP+FN);
else
   T1  = cm(cl==classes(1), cl==classes(1));
   T2  = cm(cl==classes(2), cl==classes(2));
   T3  = cm(cl==classes(3), cl==classes(3));
   F12 = cm(cl==classes(2), cl==classes(1));
   F13 = cm(cl==classes(3), cl==classes(1));
   F21 = cm(cl==classes(1), cl==classes(2));
   F23 = cm(cl==classes(3), cl==classes(2));
   F31 = cm(cl==classes(1), cl==classes(3));
   F32 = cm(cl==classes(2), cl==classes(3));
   sn = zeros(1,length(classes));
   sn(1) = T1/(T1+F21+F31);
   sn(2) = T2/(T2+F12+F32);
   sn(3) = T3/(T3+F13+F23);
   sp = 0;
   Accuracy = (T1+T2+T3)/(T1+T2+T3+F12+F13+F21+F23+F31+F32);
end
```

Haberman_import.m:

```
%% Import data from text file.
filename = 'haberman.data';
```

```
delimiter = ',';
%% Format for each line of text:
formatSpec = '%f%f%f%f%[^\n\r]';
%% Open the text file.
fileID = fopen(filename,'r');
%% Read columns of data according to the format.
dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter, …
'TextType', 'string',  'ReturnOnError', false);
%% Close the text file.
fclose(fileID);
%% Create output variable
haberman = table(dataArray{1:end-1}, 'VariableNames', …
{'age','year','nodes','class_label'});
%% Clear temporary variables
clearvars filename delimiter formatSpec fileID dataArray ans;
predictorNames = {'age', 'year', 'nodes'};
close all
knncvAcc = [];
treecvAcc = [];
svmcvAcc = [];
discvAcc = [];
gowacvAcc = [];
gowaAccu = [];
knnAcc = [];
treeAcc = [];
svmAcc = [];
disAcc = [];
knn_fitness = [];
tree_fitness = [];
svm_fitness = [];
dis_fitness = [];
gowa_fitness = [];
for r=1:6
%% Create training and validation data partition.
% 30% of data is left out for validation the generated classified
% 70% of data is used for training the classifier and for cross-vali-
dation
% rgn() is called first for reproducible results in different runs
rng(r*16);
c1 = cvpartition(haberman.class_label,'HoldOut', 0.3);
testData = haberman(c1.test, :);
test_predictors = testData(:, predictorNames);
ClassData = haberman(c1.training, :);
predictors = ClassData(:, predictorNames);
%% Create partition to be used for crossvalidation for each classifier
rng(1);
cv = cvpartition(ClassData.class_label,'KFold',10);
%% set fitness function weights
fitness_weights = [0.5 0.25 0.25];
classes = [1 2];

%% Optimize and train K-nearest neighbor classifier
distances = {'cosine', 'euclidean', 'jaccard', 'chebychev', 'seuclide-
an'};
numNeighbors = [10:30];
[knnmdl, cvknnmdl, knncvAccuracy] = knn_classifier(ClassData, cv, pre-
dictors, classes, fitness_weights, ...
                                    distances, numNeighbors);
[knnPredictions, knnscore] = kfoldPredict(cvknnmdl);
knncvAcc(r) = knncvAccuracy;
%% Optimize and train Decision tree classifier
```

```
[treemdl, cvtreemdl, treecvAccuracy] = tree_classifier(ClassData, cv,
predictors, classes, fitness_weights);
[treePredictions, treescore] = kfoldPredict(cvtreemdl);
treecvAcc(r) = treecvAccuracy;
%% Optimize and train SVM classifier
doHyperOpt = 1;
[svmmdl, cvsvmmdl, svmcvAccuracy] = svm_classifier(ClassData, cv, pre-
dictors, classes, ...
                                        fitness_weights, doHyperOpt);
[svmPredictions, svmscore] = kfoldPredict(cvsvmmdl);
svmcvAcc(r) = svmcvAccuracy;
%% Optimize and train Discriminant Analysis classifier
discriminant_types = {'linear', 'pseudolinear', 'diaglinear', 'quad-
ratic', 'pseudoquadratic', 'diagquadratic'};
[dismdl, cvdismdl, discvAccuracy] = discriminant_classifier(ClassData,
… cv, predictors, classes, fitness_weights, discriminant_types);
[disPredictions, disscore] = kfoldPredict(cvdismdl);
discvAcc(r) = discvAccuracy;
%% GOWA
lambda = 2.0;
%lambda= [0.1:0.1:5];
v=[1:3];c=4;
p = [0.1:0.25:4]; % p parameter range
m = [0.25:0.25:5]; % alpha parameter range
if (r==6)
   pl=1;
else
   pl=0;
end
%for i=1:length(lambda)
%[gowaAcc(i), gowaPredictions, gowasn(i), gowasp(i), gowatruelabels,
gowacvfitness, cvpredictions, truecvlabels, ...
% gowafitness_f] = gowa_classifier(ClassData, cv, testData, classes,…
%          fitness_weights, v, c, p, m, lambda(i), pl);
%end
[gowaAcc, gowaPredictions, gowasn, gowasp, gowatruelabels, …
gowacvfitness, cvpredictions, truecvlabels, gowafitness_f] = …
gowa_classifier(ClassData, cv, testData, classes, fitness_weights, v,
… c, p, m, lambda, pl);
gowa_fitness(r) = gowafitness_f;
gowacvAcc(r) = gowacvfitness;
gowaAccu(r) = gowaAcc;
%figure
%subplot(1,3,1)
%plot(lambda,gowaAcc)
%title('Gowa accuracies');
%ylabel('accuracy');
%xlabel('lambda');
%subplot(1,3,2)
%plot(lambda,gowasn)
%title('Gowa sensitivity');
%ylabel('sn');
%xlabel('lambda');
%subplot(1,3,3)
%plot(lambda,gowasp)
%title('Gowa specificity');
%ylabel('sp');
%xlabel('lambda');

%% plot confusion matrixes for each classifier for cross validated
models
if (r==6)
```

```
plot_cv_confusion(knnPredictions, treePredictions, svmPredictions,
disPredictions, cvpredictions, ClassData, truecvlabels);
end
%% Evaluate each classifier with test data (30% of data separated in
the beginning and not used for training)
plot_ch = 0;
%knn
[knntestPredictions, knntestscore] = predict(knnmdl, test_predictors);
[knnAccuracy, knnsn, knnsp] = evaluate_classifier(knntestPredictions,…
testData.class_label, classes, plot_ch, …
'Confusion chart for KNN classifier');
knnAcc(r) = knnAccuracy;
% tree
[treetestPredictions, treetestscore] = predict(treemdl, …
test_predictors);
[treeAccuracy, treesn, treesp] = evaluate_classifier( …
treetestPredictions, testData.class_label, classes, plot_ch, …
'Confusion chart for Tree classifier');
treeAcc(r) = treeAccuracy;
% svm
[svmtestPredictions, svmtestscore] = predict(svmmdl, test_predictors);
[svmAccuracy, svmsn, svmsp] = evaluate_classifier(svmtestPredictions,…
testData.class_label, classes, plot_ch, …
'Confusion chart for SVM classifier');
svmAcc(r) = svmAccuracy;
% dis
[distestPredictions, distestscore] = predict(dismdl, test_predictors);
[disAccuracy, dissn, dissp] = evaluate_classifier(distestPredictions,…
 testData.class_label, classes, plot_ch, …
 'Confusion chart for Discriminant Analysis classifier');
disAcc(r) = disAccuracy;
if (r==6)
plot_confusion(knntestPredictions, treetestPredictions, …
svmtestPredictions, distestPredictions, gowaPredictions, ...
testData.class_label, gowatruelabels);
end

%% calculate fitness functions for each classifier
knnfit  = calc_fit(knnAccuracy, knnsn, knnsp, fitness_weights);
treefit = calc_fit(treeAccuracy, treesn, treesp, fitness_weights);
svmfit  = calc_fit(svmAccuracy, svmsn, svmsp, fitness_weights);
disfit  = calc_fit(disAccuracy, dissn, dissp, fitness_weights);

knn_fitness  = [knn_fitness knnfit];
tree_fitness = [tree_fitness treefit];
svm_fitness  = [svm_fitness svmfit];
dis_fitness  = [dis_fitness disfit];

if (r==6)
      knnfitness  = knnfit;
      treefitness = treefit;
      svmfitness  = svmfit;
      disfitness  = disfit;
%% create a table of the results
sp = [knnsp; treesp; svmsp; dissp; gowasp];
sn = [knnsn; treesn; svmsn; dissn; gowasn];
var_names = {'sp', 'sn', 'accuracy', 'cross_val_acc', 'fitness'};
row_names = {'knn','tree','svm', 'discriminant', 'gowa'};
cvaccu = [knncvAccuracy; treecvAccuracy; svmcvAccuracy; discvAccuracy;
gowacvfitness];
accu = [knnAccuracy; treeAccuracy; svmAccuracy; disAccuracy; gowaAcc];
```

```
fitness = [knnfitness; treefitness; svmfitness; disfitness; gowafit-
ness_f]; %gowafitness];
T = table(sp, sn, accu, cvaccu, fitness, 'VariableNames', var_names,
...
    'RowNames', row_names)

%% plot ROC curves for all referenve classifiers
[knnX, knnY, knnT,  knnAUC]  = perfcurve(testData.class_label,knntest-
score(:,knnmdl.ClassNames==2), 2);
[treeX,treeY,treeT, treeAUC] = perfcurve(testData.class_label,treet-
estscore(:,treemdl.ClassNames==2), 2);
[svmX, svmY, svmT,  svmAUC]  = perfcurve(testData.class_label,svmtest-
score(:,svmmdl.ClassNames==2), 2);
[disX, disY, disT,  disAUC]  = perfcurve(testData.class_la-
bel,distestscore(:,dismdl.ClassNames==2), 2);

figure
hold on
plot(knnX,knnY)
plot(treeX,treeY)
plot(svmX,svmY)
plot(disX,disY)
hold off
legend('knn', 'tree', 'svm', 'discr','Location','SE');
xlabel('False positive rate'); ylabel('True positive rate');
title('ROC for reference classifiers');
end
end

%% calculate mean values for the 6 runs with different random data
partition
knnmeanAccuracy  = mean(knnAcc);
treemeanAccuracy = mean(treeAcc);
svmmeanAccuracy  = mean(svmAcc);
dismeanAccuracy  = mean(disAcc);
gowameanAccuracy = mean(gowaAccu);
knncvmeanAccuracy  = mean(knncvAcc);
treecvmeanAccuracy = mean(treecvAcc);
svmcvmeanAccuracy  = mean(svmcvAcc);
discvmeanAccuracy  = mean(discvAcc);
gowacvmeanAccuracy = mean(gowacvAcc);

%% Create a table for results of 10 differente runs
knnf  = [round(knn_fitness(1),3); round(knn_fitness(2),3);  …
         round(knn_fitness(3),3); round(knn_fitness(4),3); …
         round(knn_fitness(5),3); round(knn_fitness(6),3); ...
         round(mean(knn_fitness),3); round(knnmeanAccuracy,4)];
treef = [round(tree_fitness(1),3); round(tree_fitness(2),3); …
         round(tree_fitness(3),3); round(tree_fitness(4),3); …
         round(tree_fitness(5),3); round(tree_fitness(6),3); ...
        round(mean(tree_fitness),3); round(treemeanAccuracy,4)];
svmf  = [round(svm_fitness(1),3); round(svm_fitness(2),3); …
        round(svm_fitness(3),3); round(svm_fitness(4),3); …
        round(svm_fitness(5),3); round(svm_fitness(6),3); ...
        round(mean(svm_fitness),3); round(svmmeanAccuracy,4)];
disf  = [round(dis_fitness(1),3); round(dis_fitness(2),3);  …
         round(dis_fitness(3),3); round(dis_fitness(4),3); …
        round(dis_fitness(5),3); round(dis_fitness(6),3); ...
        round(mean(dis_fitness),3); round(dismeanAccuracy,4)];
gowaf = [round(gowa_fitness(1),3); round(gowa_fitness(2),3); …
         round(gowa_fitness(3),3); round(gowa_fitness(4),3); …
         round(gowa_fitness(5),3); round(gowa_fitness(6),3); ...
```

```
        round(mean(gowa_fitness),3); round(gowameanAccuracy,4)];
row_names = {'fit1','fit2','fit3','fit4','fit5','fit6',…
             'mean_fit', 'mean_accuracy'};
var_names = {'knn', 'tree', 'svm', 'discriminant', 'gowa'};
T = table(knnf, treef, svmf, disf, gowaf,  ...
          'VariableNames', var_names, 'RowNames', row_names)
```

thyroid_import.m:

```
%% Import data from text file.
filename = 'C:\Users\Salla\Documents\Gradu_matlab\new-thyroid.data';
delimiter = ',';
% Format for each line of text:
formatSpec = '%f%f%f%f%f%f%[^\n\r]';
% Open the text file.
fileID = fopen(filename,'r');
% Read columns of data according to the format.
dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter,
'TextType', 'string',  'ReturnOnError', false);
% Close the text file.
fclose(fileID);
% create thyroid data table
thyroid = table(dataArray{1:end-1}, 'VariableNames', {'class_la-
bel','T3','thyroxin','trii', 'TSH', 'TSH_diff'});
% Clear temporary variables
clearvars filename delimiter formatSpec fileID dataArray ans;
predictorNames = {'T3','thyroxin','trii', 'TSH', 'TSH_diff'};
close all
%% Initialize variables
knncvAcc = [];
treecvAcc = [];
svmcvAcc = [];
discvAcc = [];
gowacvAcc = [];
gowaAccu = [];
knnAcc = [];
treeAcc = [];
svmAcc = [];
disAcc = [];
knn_distances = [];

%% all classifiers are trained 6 times, with different random data
partition
for r=1:6
   %% Create training and validation data partition.
   % 30% of data is left out for validation the generated classified
   % 70% of data is used for training the classifier and for
   %crossvalidation. rgn(1) is called first for reproducible results
   % between different runs
   rng(r*45);
   c1 = cvpartition(thyroid.class_label,'HoldOut', 0.3);
   testData = thyroid(c1.test, :);
   test_predictors = testData(:, predictorNames);
   ClassData = thyroid(c1.training, :);
   predictors = ClassData(:, predictorNames);

   %% Create partition to be used for cross-validation
   %% for each classifier
   rng(1);
   cv = cvpartition(ClassData.class_label,'KFold',4);

   %% set fitness function weights and define used classes
```

```
fitness_weights = [0.55 0.15 0.15 0.15];
classes = [1 2 3];

%% Optimize and train K-nearest neighbor classifier
distances = {'cosine', 'euclidean', 'jaccard', 'chebychev',…
            'seuclidean'};
numNeighbors = [2:25];
[knnmdl, cvknnmdl, knncvAccuracy] = knn_classifier(ClassData, cv,…
    predictors, classes, fitness_weights, distances, numNeighbors);
[knnPredictions, knnscore] = kfoldPredict(cvknnmdl);
knncvAcc(r) = knncvAccuracy;

%% Optimize and train Decision tree classifier
[treemdl, cvtreemdl, treecvAccuracy] = tree_classifier(ClassData,…
  cv, predictors, classes, fitness_weights);
[treePredictions, treescore] = kfoldPredict(cvtreemdl);
treecvAcc(r) = treecvAccuracy;

%% Optimize and train SVM classifier
[svmmdl, cvsvmmdl, svmcvAccuracy] = multisvm_classifier(ClassData,…
  cv, predictors, classes, fitness_weights);
[svmPredictions, svmscore] = kfoldPredict(cvsvmmdl);
svmcvAcc(r) = svmcvAccuracy;

%% Optimize and train Discriminant Analysis classifier
discriminant_types = {'pseudolinear', 'diaglinear',…
                      'pseudoquadratic', 'diagquadratic'};
[dismdl, cvdismdl, discvAccuracy] = …
  discriminant_classifier(ClassData, cv, predictors, ...
                    classes, fitness_weights, discriminant_types);
[disPredictions, disscore] = kfoldPredict(cvdismdl);
discvAcc(r) = discvAccuracy;

%% Gowa
c=1;     %class label column number
v=[2:6]; % column numbers for predictors
% plot mean accuracies only for the last round
if (r==6)
    pl=1;
else
    pl=0;
end
% optimum lambda value is searched separately to shorten
% the simulation time. The codes for testing different lambda
% values are commented out here, best lambda value is determined
% based on the plots
%lambda= [0.1:0.1:5];
lambda = 2.0;
p = [1.0:0.25:8];
alpha = [0.5:0.25:8];
[gowaAcc, gowaPredictions, gowasn, gowasp, gowatruelabels, …
 gowacvfitness, cvpredictions, truecvlabels, gowafitness_f] =…
 gowa_classifier(ClassData, cv, testData, classes,fitness_weights,…
                 v, c, p, alpha, lambda, pl);
%for i=1:length(lambda)
%   [gowaAcc(i), gowaPredictions, gowasn, gowasp, gowatruelabels,…
%   gowacvfitness, cvpredictions, truecvlabels, ...
%                 gowafitness_f] = gowa_classifier(ClassData, cv,…
% testData, classes, ...
%                 fitness_weights, v, c, p, alpha, lambda(i), 0);
%   sn1(i) = gowasn(1);
```

```
%   sn2(i) = gowasn(2);
%   sn3(i) = gowasn(3);
%end
gowacvAcc(r) = gowacvfitness;
gowaAccu(r) = gowaAcc;

%% plot confusion matrixes for each classifier for cross validated
%% models, plot only results of one of the six runs
if (r==6)
   plot_cv_confusion(knnPredictions, treePredictions, …
    svmPredictions, disPredictions, cvpredictions, ClassData,…
    truecvlabels);
end

%% Evaluate each classifier with test data (30% of data separated
%% in the beginning and not used for training)
plot_ch = 0;

%knn
[knntestPredictions, knntestscore] = predict(knnmdl, …
    test_predictors);
[knnAccuracy, knnsn, knnsp] = evaluate_classifier(…
    knntestPredictions, testData.class_label, classes, plot_ch,…
    'Confusion chart for KNN classifier');
knnAcc(r) = knnAccuracy;

% tree
[treetestPredictions, treetestscore] = predict(treemdl,…
 test_predictors);
[treeAccuracy, treesn, treesp] = evaluate_classifier(…
 treetestPredictions, testData.class_label, classes,…
 plot_ch, 'Confusion chart for Tree classifier');
treeAcc(r) = treeAccuracy;

% svm
[svmtestPredictions, svmtestscore] = predict(svmmdl,…
 test_predictors);
[svmAccuracy, svmsn, svmsp] = evaluate_classifier(…
 svmtestPredictions, testData.class_label, classes, plot_ch,…
 'Confusion chart for SVM classifier');
svmAcc(r) = svmAccuracy;

% dis
[distestPredictions, distestscore] = predict(dismdl,…
 test_predictors);
[disAccuracy, dissn, dissp] = evaluate_classifier(…
 distestPredictions, testData.class_label, classes, plot_ch, …
 'Confusion chart for Discriminant Analysis classifier');
disAcc(r) = disAccuracy;

if (r==6)
   plot_confusion(knntestPredictions, treetestPredictions,…
                  svmtestPredictions, distestPredictions, ...
                  gowaPredictions, testData.class_label,…
                  gowatruelabels);
end

%% calculate fitness functions for each classifier
if (r==6)
   knnfitness  = calc_fit_multiclass(knnAccuracy, knnsn(1),…
                 knnsn(2), knnsn(3), fitness_weights);
```

```
        treefitness = calc_fit_multiclass(treeAccuracy, treesn(1),…
                      treesn(2), treesn(3), fitness_weights);
        svmfitness  = calc_fit_multiclass(svmAccuracy, svmsn(1),…
                      svmsn(2), svmsn(3), fitness_weights);
        disfitness  = calc_fit_multiclass(disAccuracy, dissn(1),…
                      dissn(2), dissn(3), fitness_weights);
        % create vectors for the result table
        sp = [knnsp; treesp; svmsp; dissp; gowasp];
        sn1 = [knnsn(1); treesn(1); svmsn(1); dissn(1); gowasn(1)];
        sn2 = [knnsn(2); treesn(2); svmsn(2); dissn(2); gowasn(2)];
        sn3 = [knnsn(3); treesn(3); svmsn(3); dissn(3); gowasn(3)];

    end
end % for r=1:6

%% calculate mean values for the 6 runs with different random data
partition
knnmeanAccuracy  = mean(knnAcc);
treemeanAccuracy = mean(treeAcc);
svmmeanAccuracy  = mean(svmAcc);
dismeanAccuracy  = mean(disAcc);
gowameanAccuracy = mean(gowaAccu);
knncvmeanAccuracy  = mean(knncvAcc);
treecvmeanAccuracy = mean(treecvAcc);
svmcvmeanAccuracy  = mean(svmcvAcc);
discvmeanAccuracy  = mean(discvAcc);
gowacvmeanAccuracy = mean(gowacvAcc);

%% Create a table of the results
var_names = {'sn1', 'sn2', 'sn3', 'accuracy', 'cross_val_accuracy',…
             'fitness'};
row_names = {'knn','tree','svm', 'discriminant', 'gowa'};
accu    = [knnAccuracy; treeAccuracy; svmAccuracy; disAccuracy;…
           gowaAcc];
cv_accu = [knncvAccuracy; treecvAccuracy; svmcvAccuracy; …
           discvAccuracy; gowacvfitness];
fitness = [knnfitness; treefitness; svmfitness; disfitness; …
           gowafitness_f];
T = table(sn1, sn2, sn3, accu, cv_accu, fitness, 'VariableNames',…
          var_names, 'RowNames', row_names)

%% Create a table for results of 6 differente runs
acc1 = [knnAcc(1); treeAcc(1); svmAcc(1); disAcc(1); gowaAccu(1)];
acc2 = [knnAcc(2); treeAcc(2); svmAcc(2); disAcc(2); gowaAccu(2)];
acc3 = [knnAcc(3); treeAcc(3); svmAcc(3); disAcc(3); gowaAccu(3)];
acc4 = [knnAcc(4); treeAcc(4); svmAcc(4); disAcc(4); gowaAccu(4)];
acc5 = [knnAcc(5); treeAcc(5); svmAcc(5); disAcc(5); gowaAccu(5)];
acc6 = [knnAcc(6); treeAcc(6); svmAcc(6); disAcc(6); gowaAccu(6)];
meanaccus = [knnmeanAccuracy; treemeanAccuracy; svmmeanAccuracy; …
             dismeanAccuracy; gowameanAccuracy];
var_names = {'accuracy1', 'accuracy2', 'accuracy3', 'accuracy4', …
             'accuracy5', 'accuracy6', 'mean_accuracy'};
row_names = {'knn','tree','svm', 'discriminant', 'gowa'};
T = table(acc1, acc2, acc3, acc4, acc5, acc6, meanaccus, ...
          'VariableNames', var_names, 'RowNames', row_names)
```

credit_import.m:

```
 %% Import data from spreadsheet
%% Setup the Import Options
```

```
opts = spreadsheetImportOptions("NumVariables", 24);
% Specify sheet and range
opts.Sheet = "Data";
opts.DataRange = "B3:Y30002";
% Specify column names and types
opts.VariableNames = ["givencredit","gender","education",…
"maritalstatus","age","past1","past2","past3","past4","past5","past6",
"billsep","billaug","billjul","billjun","billmay","billapr","prev1",
"prev2","prev3","prev4", "prev5", "prev6","class_label"];
opts.VariableTypes = ["double", "double", "double", "double", "dou-
ble", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double"];

% Import the data
clients = readtable("defaultOfCreditCardClients.xls", opts,…
                    "UseExcel", false);

observation_count = height(clients)
%find out the separate target classes
testClasses = clients.class_label;
[classes,~,idx] = unique(testClasses);
classes
%calculate how any instances there are in each class
nCount = accumarray(idx(:),1)

%% Clear temporary variables
clear opts
predictorNames = {'givencredit','gender','education','marital-
sta-
tus','age','past1','past2','past3','past4','past5','past6','billsep','
billaug','billjul','billjun','billmay','billapr','prev1','prev2',
'prev3', 'prev4', 'prev5','prev6'};
%% Create training and validation data partition.
% 30% of data is left out for validation the generated classified
% 70% of data is used for training the classifier and for crossvalida-
tion
% rgn(1) is called first for reproducible results in different runs
rng(1);
c1 = cvpartition(clients.class_label,'HoldOut', 0.4);
testData = clients(c1.test, :);
test_predictors = testData(:, predictorNames);
ClassData = clients(c1.training, :);
predictors = ClassData(:, predictorNames);

ClassData_count = height(ClassData)
cs = ClassData.class_label;
[classes,~,idx] = unique(cs);
nCount = accumarray(idx(:),1)
model_data_class0 = nCount(1)/ClassData_count
model_data_class1 = nCount(2)/ClassData_count

testData_count = height(testData)
cs = testData.class_label;
[classes,~,idx] = unique(cs);
nCount = accumarray(idx(:),1)
testdata_class0 = nCount(1)/testData_count
testdata_class1 = nCount(2)/testData_count

%% Create partition to be used for crossvalidation for each classifier
rng(1);
```

```
cv = cvpartition(ClassData.class_label,'KFold',10);

%% set fitness function weights
fitness_weights = [0.5 0.35 0.15];
classes = [0 1];

%% Optimize and train K-nearest neighbor classifier
tic
distances = {'cosine', 'euclidean', 'jaccard', 'chebychev', 'seuclide-
an'};
numNeighbors = [10:40];
[knnmdl, cvknnmdl, knncvAccuracy] = knn_classifier(ClassData, cv,…
       predictors, classes, fitness_weights, distances, numNeighbors);
[knnPredictions, knnscore] = kfoldPredict(cvknnmdl);
timeknn = toc;

%% Optimize and train Decision tree classifier
tic
[treemdl, cvtreemdl, treecvAccuracy] = tree_classifier(ClassData, cv,…
                      predictors, classes, fitness_weights);
[treePredictions, treescore] = kfoldPredict(cvtreemdl);
timetree = toc;
%% Optimize and train SVM classifier
tic
doHyperOpt = 0;
[svmmdl, cvsvmmdl, svmcvAccuracy] = svm_classifier(ClassData, cv, …
             predictors, classes, fitness_weights, doHyperOpt);
[svmPredictions, svmscore] = kfoldPredict(cvsvmmdl);
timesvm = toc;
%% Optimize and train Discriminant Analysis classifier
tic
discriminant_types = {'linear', 'pseudolinear', 'diaglinear', 'quad-
ratic', 'pseudoquadratic', 'diagquadratic'};
[dismdl, cvdismdl,discvAccuracy] = discriminant_classifier(ClassData,…
 cv, predictors, classes, fitness_weights, discriminant_types);
[disPredictions, disscore] = kfoldPredict(cvdismdl);
timedis = toc;
%% Gowa
%lambda = 2.0;
c=24;
v=[1:23];
lambda= 3.4; %
%lambda = [0.1:0.3:5];
p = [0.25:0.25:5]; % p parameter range
alpha = [0.25:0.25:5];
tic
[gowaAcc, gowaPredictions, gowasn, gowasp, gowatruelabels,…
         gowacvfitness, cvpredictions, truecvlabels, gowafitness_f] =…
         gowa_classifier(ClassData, cv, testData, [1:2], ...
         fitness_weights, v, c, p, alpha, lambda, 1);
%for i=1:length(lambda)
%[gowaAcc(i), gowaPredictions, gowasn(i), gowasp(i), gowatruelabels,
gowacvfitness, cvpredictions, truecvlabels, ...
%          gowafitness_f] = gowa_classifier_credit(ClassData, cv,
testData, [1:2], ... %_credit(ClassData, cv, testData, classes, ...
%          fitness_weights, v, c, p, alpha, lambda(i), 1);
%end
timegowa = toc;

%% plot confusion matrixes for each classifier for cross validated
models
```

```
plot_cv_confusion(knnPredictions, treePredictions, svmPredictions,…
disPredictions, cvpredictions, ClassData, truecvlabels);

%% Evaluate each classifier with test data (40% of data separated in
the beginning and not used for training)
plot_ch = 0;
%knn
[knntestPredictions, knntestscore] = predict(knnmdl, test_predictors);
[knnAccuracy,knnsn,knnsp] = evaluate_classifier(knntestPredictions,…
   testData.class_label, classes, plot_ch,…
   'Confusion chart for KNN classifier');

% tree
[treetestPredictions, treetestscore] = predict(treemdl, …
                                        test_predictors);
[treeAccuracy, treesn, treesp] = evaluate_classifier(…
                  treetestPredictions, testData.class_label, classes,…
                 plot_ch, 'Confusion chart for Tree classifier');

% svm
[svmtestPredictions, svmtestscore] = predict(svmmdl, test_predictors);
[svmAccuracy, svmsn, svmsp] = evaluate_classifier(svmtestPredictions,…
            testData.class_label, classes, plot_ch, …
             'Confusion chart for SVM classifier');

% dis
[distestPredictions, distestscore] = predict(dismdl, test_predictors);
[disAccuracy, dissn, dissp] = evaluate_classifier(distestPredictions,…
           testData.class_label, classes, plot_ch,…
         'Confusion chart for Discriminant Analysis classifier');

plot_confusion(knntestPredictions, treetestPredictions, …
      svmtestPredictions, distestPredictions, gowaPredictions, ...
               testData.class_label, gowatruelabels);

%% calculate fitness functions for each classifier
knnfitness  = calc_fit(knnAccuracy,  knnsn,  knnsp,  fitness_weights);
treefitness = calc_fit(treeAccuracy, treesn, treesp, fitness_weights);
svmfitness  = calc_fit(svmAccuracy,  svmsn,  svmsp,  fitness_weights);
disfitness  = calc_fit(disAccuracy,  dissn,  dissp,  fitness_weights);
gowafitness = calc_fit(gowaAcc, gowasn, gowasp, fitness_weights);

%% create a table of the results
sn = [knnsn; treesn; svmsn; dissn; gowasn];
sp = [knnsp; treesp; svmsp; dissp; gowasp];
var_names = {'sp', 'sn', 'accuracy', 'cross_val_accuracy', 'fitness'};
row_names = {'knn','tree','svm', 'discriminant', 'gowa'};
accu = [knnAccuracy; treeAccuracy; svmAccuracy; disAccuracy; gowaAcc];
cv_accu = [knncvAccuracy; treecvAccuracy; svmcvAccuracy; discvAccu-
racy; gowacvfitness];
fitness = [knnfitness; treefitness; svmfitness; disfitness; …
           gowafitness];
T = table(sp, sn, accu, cv_accu, fitness, 'VariableNames', var_names,…
          'RowNames', row_names)

%% training times
traintimes = [round(timeknn/60,1); round(timetree/60,1);
round(timesvm/60,1); round(timedis/60,1); round(timegowa/60,1)];
t = table(traintimes, 'VariableNames', "training_time_in_minutes", …
          'RowNames', row_names)
```

```
%% plot ROC curves for all classifiers
[knnX, knnY, knnT,  knnAUC] = perfcurve(testData.class_label, …
                              knntestscore(:,knnmdl.ClassNames==1), 1);
[treeX,treeY,treeT, treeAUC] = perfcurve(testData.class_label,…
                            treetestscore(:,treemdl.ClassNames==1), 1);
[svmX, svmY, svmT,  svmAUC]  = perfcurve(testData.class_label,…
                              svmtestscore(:,svmmdl.ClassNames==1), 1);
[disX, disY, disT,  disAUC]  = perfcurve(testData.class_label,…
                              distestscore(:,dismdl.ClassNames==1), 1);

figure
hold on
plot(knnX,knnY)
plot(treeX,treeY)
plot(svmX,svmY)
plot(disX,disY)
hold off
legend('knn', 'tree', 'svm', 'discr', 'Location','SE');
xlabel('False positive rate'); ylabel('True positive rate');
title('ROC for reference classifiers for credit default data');
```

Codes for functions init_data, ideal_vectors and calcfit, which are used in gowa_classifier, are available on request from the original author Pasi Luukka (pasi.luukka@lut.fi).