

UTRECHT UNIVERSITY

MASTER'S THESIS

Machine learning dissected

Author:
Raj JAGESAR

Supervisors:
Dr. Marco SPRUIT
Prof. dr. Sjaak BRINKKEMPER
Dr. Martien KAS
Dr. Jacob VORSTMAN

*A thesis submitted in fulfillment of the requirements
for the degree of Master of Science*

in the

Center for Organization and Information
Department of Information and Computing Sciences

May 19, 2016

Declaration of Authorship

I, Raj JAGESAR, declare that this thesis titled, “Machine learning dissected” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at Utrecht University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"I keep saying that the sexy job in the next 10 years will be statisticians. And I'm not kidding."

Hal Varian, Chief Economist at Google (2009)

UTRECHT UNIVERSITY

Abstract

Faculty of Science
Department of Information and Computing Sciences

Master of Science

Machine learning dissected

by Raj JAGESAR

The research presented in this thesis addresses machine learning techniques and their application in the context of classification problems. Furthermore as this thesis is centered around a medical initiative (Behapp) the insights found were applied to the data produced by this initiative.

The direction of study on general machine learning techniques was chosen in order to model the knowledge on how to create optimized machine learning models. Furthermore, since it concerns the analysis of a medical data set the usage of transparent modeling techniques is preferred allowing us to relate the input (data) to the output (classification). This relates back to the goal of creating optimized models since transparent techniques are known to be outperformed by their non transparent counterparts.

Using the modeling approach by Weerd and Brinkkemper (2008) the machine learning techniques were modelled into a method in the form of a process-deliverable-diagram. The method was then applied to two datasets to evaluate the potential for improvements in performance.

We found that models generated using our method showed increased performance in terms of classification accuracy and overall reliability of the results. Next we applied transparent modeling techniques and the sociability scoring model (Eskes et al., 2016) to the data of the Behapp initiative. As expected, the in-depth look reveals various patterns where patients and controls are separated in the data.

In light of the results we feel that the method created enables further reasoning on the application of machine learning techniques in a single procedural data mining approach and may be extended to include procedures relevant to other domains. Last we find that the concept of an aggregated sociability score shows promise in expressive value having applied it to patient data for the first time.

Acknowledgements

First and foremost, I would like to thank Lord Shiva for his ongoing protection, support and guidance through life and subsequently this research project, I hope to remain on the right path for the time to come.

Next, my thanks go out to the following persons who have played a significant role in getting to his point:

- Dr. Marco Spruit, his insights, trust and enthusiasm inspired confidence leading to an overall positive experience during this research project.
- Prof. dr. Sjaak Brinkkemper for his oversight on this research project as my second supervisor.
- Prof. dr. Martien Kas and dr. Jacob Vorstman for their warm welcome to their line of research and the continuation of this fascinating project as the starting point of my career.
- My girlfriend Wiemla, her unending support and patience have allowed me to power through every challenge encountered while running through the business informatics program. Words cannot express all that she means to me.
- My buddy Bas Hovestad, it was an honor and privilege to have worked alongside him during the business informatics program. His experience with performing a research project has proven to be valuable to my own work. May our (professional) paths cross again in the future.

Raj Jagesar - 19th of May, 2016

Contents

Declaration of Authorship	iii
Abstract	vii
Acknowledgements	ix
1 Introduction	1
1.1 About machine learning	1
1.1.1 Description and prediction	2
1.1.2 Knowledge discovery and data mining	2
1.2 Knowledge discovery in medicine	2
1.3 Case study: The Behapp project	4
1.4 Problem statement & Research objective	5
1.5 Research questions	6
1.6 Research approach	7
1.7 Relevance	9
1.7.1 Scientific relevance	9
1.7.2 Social relevance	9
2 Data mining process models	11
2.1 KDD	11
2.2 Human-centered approach of data mining	12
2.3 SEMMA	12
2.4 Epicycles of Analysis	13
2.5 The Three-phases method	13
2.6 CRISP-DM	14
2.6.1 Hierarchical breakdown	14
2.7 Process model selection	16
3 Machine learning concepts	19
3.1 Data understanding	19
3.1.1 Histogram graphs & Pairwise scatterplots	20
3.1.2 Correlation plot	20
3.2 Data preparation	21
3.2.1 Feature selection	21
3.2.2 Feature construction	22
Feature extraction	22
Feature engineering	23
3.2.3 Feature transformation	23
Tresholding and discretization	24
Normalization and calibration	25
3.3 Modeling	25
3.3.1 Model types	26
Tree models	26

Linear models	27
Probalistic models	29
Rule models	30
Ensembles	30
3.3.2 Parameter optimization	31
Grid search	31
Random search	32
Bayesian optimization	32
3.4 Model evaluation	33
3.4.1 Performance measures	33
Confusion matrix based measures	33
ROC (AUC)	34
3.4.2 Resampling methods	35
Resubstitution and holdout	35
Stratified k-fold cross validation	36
Leave one out	36
Bootstrapping	36
3.5 Accuracy versus transparency	36
4 Method fragments	39
4.1 General method fragments	39
4.1.1 Data understanding	39
4.1.2 Data preparation	42
4.1.3 Modeling & Evaluation	46
4.2 Behapp	50
4.2.1 Daily average number of places visited	50
5 Method evaluation	55
5.1 Goal definition	55
5.2 Experiment design	56
5.3 Data description	57
5.3.1 Iris data set	57
5.3.2 Behapp data set	57
5.4 Data & Interpretation	58
6 Behapp data insights	59
6.1 Data wrangling	59
6.2 Data preparation - Feature selection	60
6.3 Modeling - Tree model	60
6.4 Social Profiling	61
7 Conclusion	65
7.1 Conclusion of sub-questions	65
7.2 Conclusion of main question	67
8 Discussion	69
8.1 Limitations	69
8.2 Future research	70
Bibliography	71
A The Integrated Method	77

B Behapp data set	79
C Experiment notebook - Iris data	85
D Experiment notebook - Behapp data	91
E Notebook - Behapp data wrangling	97
F ICIS 2016 Paper submission	115

List of Figures

1.1	Unmodified Eskes scoring applied to Behapp data	5
1.2	Research model	8
1.3	Design science framework by Hevner et al. (2004)	9
2.1	KDD methodology	12
2.2	Human-centered approach of datamining by Mariscal, Marbán, and Fernández (2010b)	12
2.3	SEMMA methodology (Mariscal, Marbán, and Fernández, 2010b)	13
2.4	Task overview of the Three-phases method (Vleugel, Spruit, and Daal, 2009)	14
2.5	The CRISP-DM process model lifecycle (Chapman et al., 1999)	15
2.6	Four abstraction levels of the CRISP-DM methodology . . .	15
2.7	Generic tasks in the data preparation phase	16
3.1	Histograms and pairwise scatterplots based on the Iris data set	20
3.2	Correlation plot based on the Iris data set	21
3.3	Principal component analysis applied to Iris data set	23
3.4	Decision tree model of the Iris data set	27
3.5	Overview of MLP layers	28
3.6	SVM decision boundary placement, adapted from (Flach, 2012)	29
3.7	Representation of binarized categorical data	29
3.8	ROC example based on the Iris data set	35
4.1	Process-deliverable-diagram of the data understanding phase	40
4.2	Process-deliverable-diagram of the data preparation phase .	43
4.3	Process-deliverable-diagram of the modeling & evaluation phase	47
4.4	Process-deliverable-diagram for processing location data . .	51
5.1	Graphical overview of the experiment model	56
6.1	Decision tree model of the Behapp data set	61
6.2	Sociability score plot of Behapp participants - Original Eskes	62
6.3	Sociability score plot of Behapp participants - Modified Eskes	63

List of Tables

2.1	DM process model highlights	16
2.2	Poll on data mining process model usage by KDnuggets.com	17
3.1	Datatype transformation directions by Flach (2012)	24
3.2	Example of tresholding and discretization	24
3.3	Normalization example, adapted from Flach (2012)	25
3.4	Rule model applied to the Iris data set	30
3.5	Searchgrid example	31
3.6	Confusion matrix example, adapted from Markham (2014) .	33
4.1	Activity table of the data understanding phase	40
4.2	Concept table of the data understanding phase	41
4.3	Activity table of the data preparation phase	44
4.4	Concept table of the data preparation phase	45
4.5	Activity table of the modeling & evaluation phase	48
4.6	Concept table of the modeling & evaluation phase	49
4.7	Activity table for processing location data	52
4.8	Concept table for processing location data	53
5.1	Classification score overview of the Iris and Behapp data set	58

List of Abbreviations

ML	Machine learning
DM	Data mining
KDD	Knowledge discovery in databases
ANN	Artificial neural network
SVM	Support vector machine
ME	Method engineering
PDD	Process-deliverable diagram
SLR	Structured literature review
SEMMA	Sample, explore, modify, model, assess
EoA	Epicycles of Analysis
3PM	Three-phases method
CRISP-DM	Cross-industry standard process for data mining
EDA	Exploratory data analysis
PCA	Principal component analysis
MLP	Multi layer perceptron with backpropagation
ARM	Association rule mining
TPR	True positive rate
TNR	True negative rate
FPR	False positive rate
FNR	False negative rate
ROC	Receiver operating characteristic
AUC	Area under the curve
GAM	Generalized additive modeling
NFLT	No free lunch theorem
WEKA	Waikato environment for knowledge analysis

Dedicated to Wiemla, the light of my life.

Chapter 1

Introduction

1.1 About machine learning

This thesis is centered around a research initiative at the University Medical Center Utrecht with the end goal of solving a classification problem on a medical dataset. In other words we will attempt to uncover underlying patterns in the data that can be used to distinguish between patients and participants from the control group.

In order to achieve this we will draw from the field of machine learning (ML). Flach (2012) defines machine learning as “the systematic study of algorithms and systems that improve their knowledge or performance with experience”. The author explains that the act of learning is similar to that of humans since ML practices generally lead to improved knowledge but not necessarily to improved performance on a task.

In this context the concept of experience can take different forms e.g. distance measures and training data from which the relative similarity between cases can be derived. These types of experience are closely related to the three types of ML tasks that are applied in practice:

- **Supervised learning:** the system is provided with training data which contains both the input and output data (labels). Based on this data the system determines which input conditions belong to a certain output, this process yields a decision model. The resulting model is then used on an unlabeled dataset to classify the unseen instances of the set (Kotsiantis, Zaharakis, and Pintelas, 2007).
- **Unsupervised learning:** the system does not know a training phase as with supervised machine learning since it is directly presented with unlabeled data. It is up to the system’s algorithm, by using distance measures, to uncover patterns in the data. An example of unsupervised learning is clustering where instances of a dataset are grouped based on commonalities in their features (Jain, Murty, and Flynn, 1999).
- **Reinforcement learning:** also known as *trial and error learning*, the system learns by interacting with an environment. During this interaction the system is provided with a signal indicating how well the system is performing. The system can choose to act based on experience or to explore a new choice (Sutton and Barto, 1998).

1.1.1 Description and prediction

The creation of ML systems spans various disciplines like mathematics, statistics and computer science. These disciplines combined support the act of learning and result in models that are fitted to data. The challenge is to derive models that are accurate in the sense that they reflect the underlying patterns in the data whilst ignoring peculiarities that do not represent reality. This is also known as *signal noise*.

The popular and well known purpose of models is to make predictions on new (and unseen) examples of data. Insurance companies, for example, have models to help predict whether insurance claims are fraudulent or not based on historic data (Phua et al., 2010).

However ML techniques are also well suited to explore the underlying patterns of a dataset. According to Lan, Frank, and Hall (2011) "people frequently use machine learning techniques to gain insight into the structure of their data rather than to make predictions for new cases". This is of particular importance to this research project since, as we will explain in section 2, medicine requires validation of results.

1.1.2 Knowledge discovery and data mining

Both in practice and in the body of literature, ML systems and techniques are often part of processes known as *knowledge discovery in databases* (KDD) and mostly *data mining* (DM). Both processes are aimed to gain knowledge from large amounts of data. Agrawal and Shafer (1996) and Fayyad, Piatetsky-Shapiro, and Smyth (1996) place DM as a formal step in the KDD process, describing DM as the iterative application of statical and logical techniques to fit or derive models from data.

However, over the years, DM has also become accepted as a term encompassing all the phases of the KDD process meaning that the terms can be used as synonyms (Mariscal, Marbán, and Fernández, 2010a). In line with this view DM is defined as the process where data is selected, explored and modeled for the purpose of knowledge discovery (Giudici, 2005). Machine learning can then be considered as one approach to performing a DM project. The domain of ML has its own techniques and guidelines that are applied during this process which impact the data preparation, modeling and knowledge discovery aspects of data mining. In practice this makes the terms closely interrelated but not completely interchangeable.

1.2 Knowledge discovery in medicine

With ongoing advancements in technology, healthcare institutions are also finding themselves in a position where large amounts of data are generated which require an automated means of knowledge extraction (Milovic, 2012). Furthermore, Bellazzi and Zupan (2008) find that care processes like prognosis, diagnosis and treatment planning could benefit from the application of predictive analytics. For example, data mining has been successfully applied in health care to help predict insurance fraud, underdiagnosed patients and classify people in health-risk categories (Yoo et al., 2012).

The body of literature repeatedly comes back to the following points of attention when data mining is applied in the medical domain. The findings are briefly reported below:

- **Results should be explainable:** in medical practice it is just as important to know why a certain conclusion is drawn by a machine learning algorithm next to the conclusion itself. A medical practitioner should be able to identify and present the underlying factor(s) that made the machine learning system classify a person as a potential sufferer from an illness. This is an important factor in the design of a machine learning system since not all machine learning algorithms offer the kind of insight that is required. For example, artificial neural networks (ANN) and support vector machine's (SVM) are praised for their accuracy and performance but are also considered black boxes since they do not provide information on how the system came to a conclusion. On the other hand decision trees and bayesian networks do provide insight into the inner workings of a decision model (Bellazzi and Zupan, 2008) (Lavrač, 1999).
- **Specifics regarding performance evaluation:** the performance of algorithms is usually measured in accuracy and speed (time). However in medicine two other measures are found to be more important and more frequently used: *sensitivity and specificity* (Lavrač, 1999). Accuracy is the proportion of cases that are correctly classified. Sensitivity is concerned with the proportion of positive cases that are correctly classified as positive, therefore it is also known as the *true positive rate*. Specificity measures the opposite from sensitivity, it is concerned with the proportion of negative cases that are correctly classified as negative (*true negative rate*). In practice one should strive to maximize the true positive rate (sensitivity) to ensure that patients with illnesses are not left untreated. The true negative rate should also be maximized to prevent misclassification costs in terms of financial cost and patient discomfort / danger due to unnecessary and possibly invasive tests (Freitas, Costa-Pereira, and Brazdil, 2007).
- **Low data quality:** the data available in the medical field is generally found to be of low quality for data mining purposes compared to other fields. This is largely attributed to the heterogeneity of medical data and the occurrence of missing values in datasets. Yoo et al. (2012) explain this by stating that in practice patients with the same illness may undergo different examinations and lab tests resulting in different datasets. Furthermore the datasets are found to be smaller with the number of instances (rows) averaging from tens to several thousands (Bellazzi and Zupan, 2008).

Last, in a comparative study of data mining methods on breast cancer survivability Delen, Walker, and Kadam (2005) conclude that the results of data mining exercises are found to be meaningless without evaluation by medical professionals with experience in the problem domain. Evaluation by a professional was found to be necessary in order to determine whether the findings were logical, actionable and / or novel. Because of this the authors do not foresee replacement of medical professionals and researchers

by machine learning systems. Instead they consider the technology as complementary to the daily practice of medical professionals where it will be used in the form of decision support systems.

1.3 Case study: The Behapp project

The Behapp project is an initiative of researchers of the translational neuroscience department at the University Medical Center Utrecht (UMCU). The end goal of this research project is to detect the onset of psychiatric disorders in a very early stage. Early detection and prevention are key since at present psychiatric disorders can only be identified once they are fully developed. However it is known that many major psychiatric disorders have a slow, gradual onset marked by subtle changes in social behaviors before the full blown manifestation emerges.

The Behapp project is currently focused on patients who have been diagnosed with schizophrenia. The concept of prevention is especially relevant to this disorder since at this point in time the disorder cannot be reversed but only treated. The disorder is usually preceded by a state of psychosis which manifests at the start of adolescence. An important warning sign for the development of this state is social withdrawal which is defined as a negative symptom in the scoring method (*PANSS*) used to formally diagnose patients as schizophrenic (Kay, Fiszbein, and Opfer, 1987). By recognizing a change in social and movement patterns the researchers aim to stop the state of psychosis taking shape preventing worse.

To explore this dimension in depth a mobile application was developed to objectively record social and movement acts of participants through their smartphones. According to Proudfoot (2013) smartphones lend themselves well for these purposes since they are becoming increasingly ubiquitous and are characterized as highly personal devices. Furthermore, most smartphones contain an array of built in sensors and connectivity options which allow for tracking of movements and social acts.

The smartphone application is named Behapp and records a host of features like e.g. the number of calls made, the number of bluetooth devices in proximity, the size of one's social circle, location data and more. The application is currently in a beta development phase and used by a control group and a patient group. The project is currently at a phase where we aim to explore whether social- and mobility patterns can be accurately correlated to mental disorders. The data gathered through the participants smartphones will be used as the dataset for analysis in this thesis. As mentioned in section 1.1 we will determine whether we can find (underlying) patterns where patients can be separated from the control group.

We will also look at the sociability scoring model created by Eskes et al. (2016), the model was created as part of an earlier master's thesis project for the Behapp initiative. The model, when applied, yields aggregated scores which factor in all the measurements that are found to be relevant to social exploration (mobility) and communication efforts of a participant. By plotting these scores on their own axes we gain a two dimensional overview of a participants social efforts. See Figure 1.1 for an example overview applied to the Behapp dataset. The plot shows the transformed data, using the sociability scoring model, of both patients and controls that have participated

in one of the studies that employed the Behapp application. The x-axis expresses the communicative efforts of the participants whereas the y-axis is an expression of the exploration efforts. Paul Eskes has named this scoring practice social profiling. The original model factors in a limited number of features, we plan on extending the model with newer and improved features.

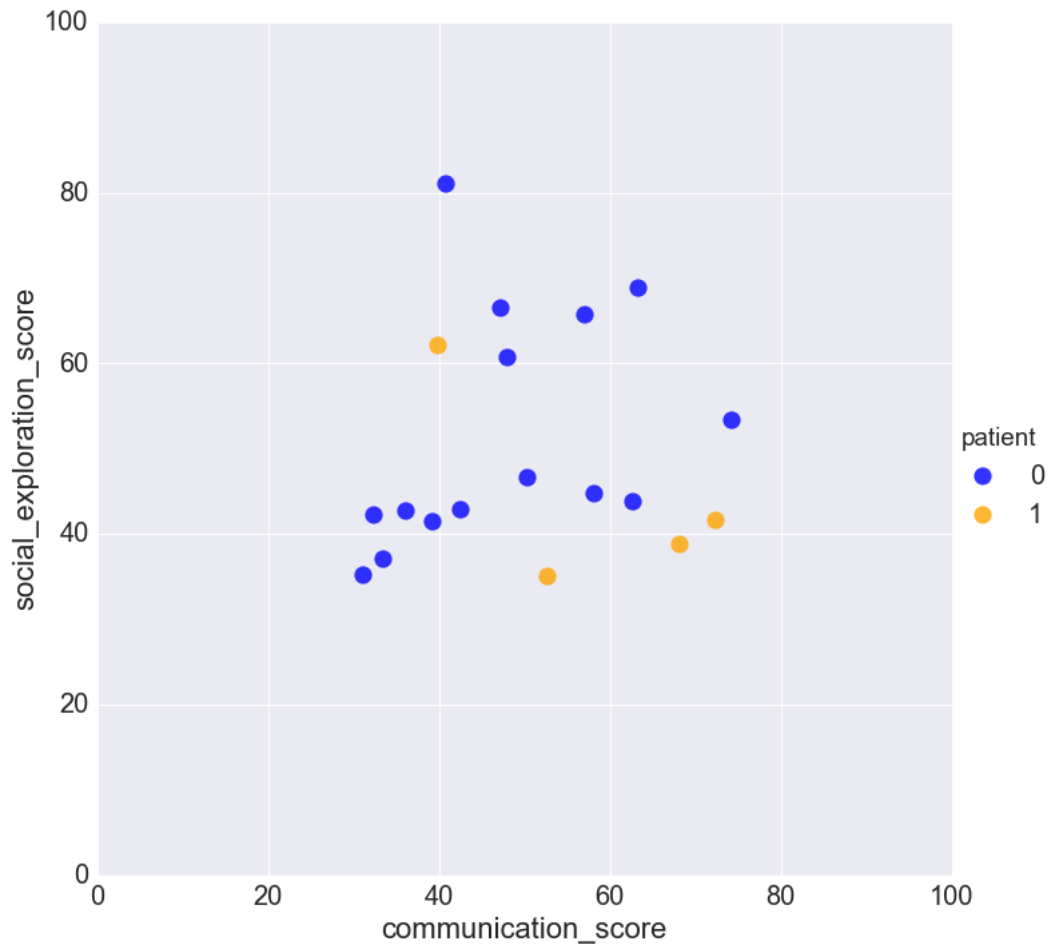


FIGURE 1.1: Unmodified Eskes scoring applied to Behapp data

1.4 Problem statement & Research objective

Despite the growing usage and popularity of machine learning techniques in data mining exercises, (correctly) applying these techniques remains a challenge. We list the main challenges below:

- The field knows many different use cases, each of which has a sizeable body of literature surrounding the specific case. The literature is usually found to be heavy on mathematical terminology and aimed at the computer science community. This prevents researchers from other fields in learning and (correctly) applying machine learning techniques in their own research (Domingos, 2012).

- In line with the aforementioned, applying machine learning techniques confronts users with many degrees of freedom in how to assemble and configure a learning system. One example of this is the fact that algorithm performance is largely determined by parameter settings, these settings are specific for each class of algorithm. However in practice end users usually do not have enough knowledge on how to find optimal parameter settings (Yoo et al., 2012). Many users leave the parameters to their default settings and base algorithm selection on reputation and / or intuitive appeal (Thornton et al., 2013). This may lead to researchers using underperforming algorithms and gaining suboptimal results.
- Concerning the creation of models: ML shows that currently there is a trade-off to be had between accuracy and transparency (Kamwa, Samantaray, and Joós, 2012a). In practice this means that algorithms which yield a high amount of insight into the data do not perform as well as their non-transparent (black box) counterparts and the other way around.

We see this as a problem where information is not available in a form that supports the researcher in walking through an optimized ML exercise. Therefore, as part of this thesis, we aim to design a deliverable which fills this need. The deliverable should guide the application of ML techniques whilst at the same time provide information on how to cope with challenges like parameter optimization and model transparency.

1.5 Research questions

By taking into account both the needs of the Behapp project as well as the generalizability of its problem space, the problem statement and research objective the main research question is formulated as follows:

- **How can a domain independent method be developed to guide the process of constructing transparent machine learning models?**

This research project and the resulting method will be scoped in line with the basic requirements of the Behapp project. Therefore the project will focus on supervised machine learning for classification tasks on structured data. Coincidentally this is one of the most applied and mature areas within the machine learning practice (Kotsiantis, Zaharakis, and Pintelas, 2007). By creating this method we intend to make a contribution to science in general by presenting a comprehensive method to generate optimal models.

We formulate the following sub questions to structure and guide our research objective:

1. *Which data mining process models are available and how do they fit machine learning techniques?*

The method will require a structure in which the ML techniques can be placed. These structures can be found in the form of various DM process models like e.g. CRISP-DM and SEMMA. We will select the

process model that satisfies our goal of presenting information in a comprehensible way.

2. *What are the main concepts involved in constructing a supervised machine learning system for classification tasks?*

An overview will be created of all the concepts that are relevant in the process of creating a supervised machine learning classification model. We will uncover and document the relationships, including conditional dependencies, between the concepts. Knowing how the concepts relate to each other will help in determining the order of the steps and their correct placement in the method.

- (a) *Which of these concepts have aspects that support (automated) configuration and optimization and how does this work?*

The method aims to ensure that researchers make full use of the possibilities to finetune their machine learning set up to achieve optimal models. Concepts that require configuration and optimization are therefore explored in-depth.

3. *What are transparent models and which techniques are suitable for this type of modelling?*

We will study the accuracy-transparency tradeoff and document how transparent models can be attained. Furthermore we will explore the state of the art in model transparency ML techniques.

4. *How can the concepts that are found be modelled into a method?*

In order to design the method we will turn to the practice of method engineering (ME). Method engineering is defined as "the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems" (Brinkkemper, 1996).

From this discipline we will apply the meta-modelling approach by Weerd and Brinkkemper (2008), this yields a process-deliverable diagram (PDD). A PDD consists of two diagrams, the left-hand side shows an UML activity diagram (processes) and the right-hand side shows an UML class diagram (concepts / deliverables). Both diagrams are integrated and display how the activities are tied to each deliverable, furthermore the activities and the concepts are each explained in separate tables.

1.6 Research approach

The research framework by Verschuren, Doorewaard, and Mellion (2010) has been adopted to illustrate the basic outlines of this research project. The framework is depicted in Figure 1.2.

First a theoretical foundation is established on the subjects of data mining, machine learning, its application in medicine and model transparency (a). From this foundation we will design general method fragments that are scoped on separate phases in the DM process. Where applicable we will also design Behapp specific method fragments. These fragments will

become part of the method base (b). Next assembled methods are evaluated on performance (c) to ensure that the research goals are satisfied. Having a method base allows the practice of method assembly where we create a project specific method by combining the general and Behapp specific method fragments (d) (Brinkkemper, Saeki, and Harmsen, 1999).

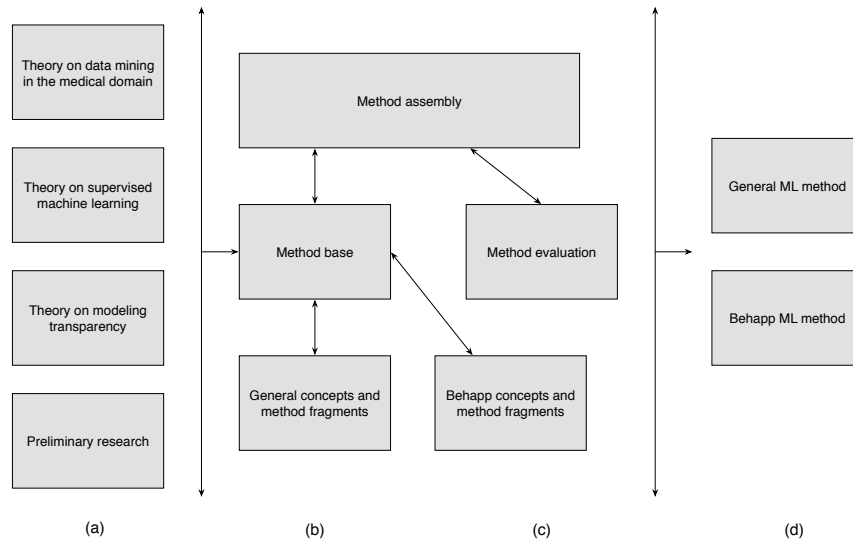


FIGURE 1.2: Research model

To establish a theoretical foundation a literature study will be performed. A preliminary look into this field has shown it to be mature but still highly expansive. Therefore, due to the expected high number of concepts that will be studied performing a structured literature review (SLR) is deemed unfeasible in the time allotted for this research project. Therefore the literature selection process will be based on reference harvesting techniques like backward chaining and forward chaining als known as *snowballing* (Jalali and Wohlin, 2012). The library of choice will be Google Scholar¹ because of its large index of scientific literature in the domain of computer science. Furthermore the library offers specific features that support and simplify the snowballing process.

Next the research process will be performed within the *design science* paradigm as defined by Hevner et al. (2004). The paradigm fits this research project since the authors state that "the design science paradigm seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts". According to the authors viable forms of these artifacts can be for example methods and models (Hevner et al., 2004). More importantly the authors provide a set of research guidelines to ensure that the artifacts are produced with scientific rigor both in construction and evaluation. Furthermore the authors state that the artifacts should serve valid business problems and be readily communicable outside of the scientific domain.

To illustrate the research project in the context of the design science paradigm the design science framework has been adopted as depicted in Figure 1.3.

¹<http://scholar.google.com/>

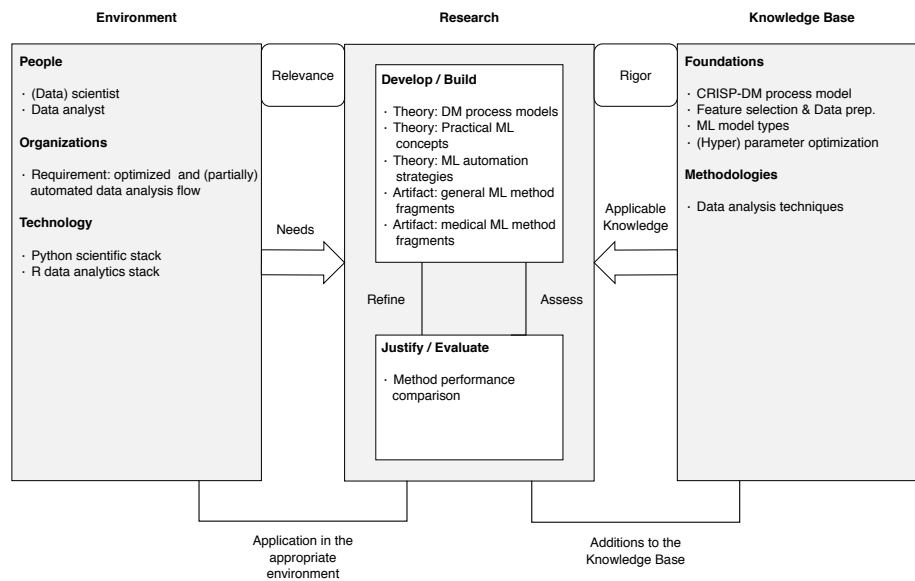


FIGURE 1.3: Design science framework by Hevner et al. (2004)

1.7 Relevance

In this section the relevancy of the project is explained and discussed from both a scientific and a social viewpoint.

1.7.1 Scientific relevance

With the steadily growing availability of data storage space and computing power advanced data mining efforts are coming in reach of more people. However as explained earlier applying machine learning techniques is not a straightforward process and requires specific know-how of many different aspects involved in the process. This research project aims to uncover these bits of know-how and integrates them into a method that can be used to generate optimal classification models.

Fellow researchers looking to apply machine learning in their work can use this method as a guide in the data mining process. Researchers that already apply machine learning in their work may look to the method for possible optimizations to their machine learning system. Furthermore the method and the underlying method fragments can be used as source material for the creation of modified method fragments to be applied in other research domains. This is demonstrated in this project as it is part of a medical initiative. Lastly, the resulting method can be used as a base to support the development of advanced data mining software.

1.7.2 Social relevance

The social relevance of this project is related to two separate goals. First, in line with the scientific relevance of this project the utility of the method extends beyond researchers. The method can also be relevant for use in businesses and industry by (aspiring) data science professionals.

Next, as this research project is part of the Behapp initiative as described in section 1, supplying the initiative with the methods to further their insight into mental disorders may one day help in the prevention, diagnosis and treatment of patients with these disorders.

Chapter 2

Data mining process models

In this chapter we will provide a short overview of frequently cited and used data mining process models. Pressman (2005) defines a process model as a set of activities including their inputs and outputs that are needed to complete a job. The goal of this chapter is to determine which process model provides a good basis for the creation of our method. We are looking for process models with the following properties:

- **Prevalence in business and academics**

Familiarity with the process model supports our goal to create a method that is easy to understand, furthermore it offers the users the benefit of only having to slightly adjust one's processes that are already in place.

- **Extended level of depth**

In line with the first requirement it is important for the process model to have a level of depth that allows to comprehend which activities belong to which phase in a data mining project.

- **Oriented around the technical aspects of a data mining exercise**

ML techniques are mainly involved with the technical aspects of a data mining exercise. A process model should therefore explicitly focus on distinguishing technical tasks in the different phases of a project.

We will explore the following DM process models: KDD, Human-centered approach of DM, SEMMA, The Three-phases method, Epicycles of Analysis and CRISP-DM.

2.1 KDD

Knowledge discovery in databases (KDD) is one of the first formalizations of data mining process models. Fayyad, Piatetsky-Shapiro, and Smyth (1996) define KDD as the process to find potentially useful and understandable patterns in data. The authors emphasize the need for proper transformation and preprocessing procedures like data selection and cleaning before starting with the actual analysis. Furthermore, although not visualized by the authors (Figure 2.1), the process model commences with a domain understanding phase in which goals of the project are acquired from the viewpoint of the customer. The phases of the process model are: domain understanding, selection, preprocessing, transformation, data mining (modeling) and interpretation / evaluation.

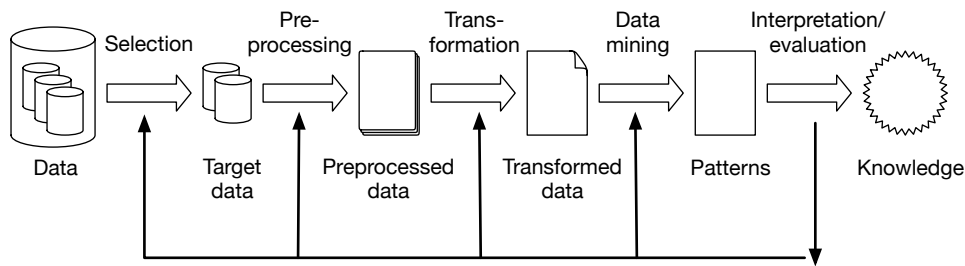


FIGURE 2.1: KDD methodology

2.2 Human-centered approach of data mining

The human-centered approach is described as a practical approach towards the KDD process (Brachman and Anand, 1996) (Gertosio and Dussauchoy, 2004). The process model is focussed on the point of view of the data analyst in the execution of the process. According to (Mariscal, Marbán, and Fernández, 2010b) this is the main differentiator of this approach compared to the general KDD process. The high level phases (Figure 2.2) are: task discovery, data discovery, data cleaning, model development, data analysis and output generation.

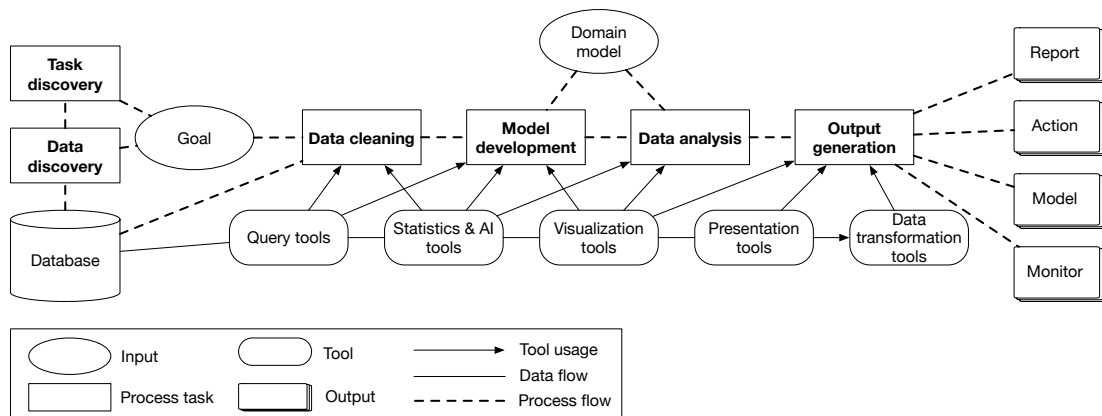


FIGURE 2.2: Human-centered approach of datamining by Mariscal, Marbán, and Fernández (2010b)

2.3 SEMMA

The SAS Institute is a software company specializing in the development of data processing and analysis software. Their flagship product SAS enterprise miner implements and adheres to the SEMMA process model which has been developed in-house. SEMMA stands for: sample, explore, modify, model and assess.

SEMMA is different from most other process models because it skips steps related to learning the application domain and exploring and evaluating the business goals of a DM project. According to Mariscal, Marbán, and Fernández (2010b) these steps are found to be essential in carrying out a succesful DM project.

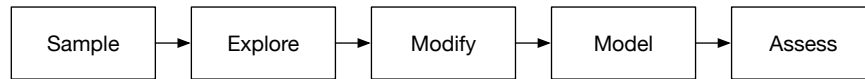


FIGURE 2.3: SEMMA methodology (Mariscal, Marbán, and Fernández, 2010b)

2.4 Epicycles of Analysis

The epicycles of analysis (EoA) process model is a cyclical model which consists of the following main phases:

- Stating the question
- Exploratory data analysis
- Model building
- Interpret
- Communicate

Peng and Matsui (2016) present this model in their educational work *The Art of Data Science*, it differs from the other process models discussed due to its emphasis on the cyclical nature of each phase. Each phase (epicycle) should follow the same pattern: 1) setting expectations, 2) collecting information & comparison with the expectations, 3) revising the expectations or fixing the data so that the data and the expectations match. This pattern essentially warrants critical thought and reasoning about data along each step of the way ensuring that no erroneous conclusions are drawn from the analysis.

2.5 The Three-phases method

The Three-phases method (3PM) is another cyclical process model that is tuned towards cases where the data mining process is outsourced to a third party (Vleugel, Spruit, and Daal, 2009). As the name suggests the model consists out of three phases. The first phase is named data retrieval and is centered around activities to understand business needs (based on hypotheses) and to prepare and cleanse data. The second phase encompasses the act of data mining itself with the addition of setting the goal for what type of model should be realised (either predictive or descriptive). Lastly, in phase three solutions are explored to embed the results in the case organization. The authors apply the *Business IT Alignment model* (BITA) by Scheper (2002) to map conclusions from the data to organizational domains (e.g. people and culture, monitoring and control) and decision levels (operational, tactical & strategic). Figure 2.4 below provides an overview of the main phases and each subtask within each phase. The labels of each task reflect the outsourcing oriented nature of the model where C stands for a task belonging to the case organization and T stands for a task belonging to the third party.

Activities of the Three-phases model								
Data retrieval		Data mining			Results implementation			
Analyze business needs	Create data set	Set goal	Perform modeling	Perform validation	Analyze business domains	Create approach	Optional: Align approaches	Deploy result
Perform business interviews (CT)	Retrieve entities and attributes (T)	Set purpose (CT)	Choose technique (T)	Validate result (CT)	Perform business interviews (CT)	Select data mining result (T)	Interview end-users / stakeholders (CT)	Business preference; either:
Review business documents (T)	Create raw data set (T)	Define goals (CT)	Performing modeling (T)	Choice (C)	Define business preference (T)	Analyze AS-IS description (T)	Create alignment approach (T)	- Create change report (T)
Define hypothesis (CT)	Apply data filters (T)	Create data view (T)	Interpret result (T)		Create AS-IS description (T)	Create TO-BE description (CT)		- Implement result (CT)
	Apply data enrichment (CT)		Document result (T)					

FIGURE 2.4: Task overview of the Three-phases method (Vleugel, Spruit, and Daal, 2009)

2.6 CRISP-DM

CRISP-DM (*CRoSS-Industry Standard Process for Data Mining*) is a methodology conceived during the late 90's by Chapman et al. (1999). Its main goal was to meet the need for a uniform approach towards data mining. The methodology consists of two process models to structure and guide data mining projects. It is well known for the process model illustrating the life cycle of a data mining project as shown in Figure 2.5.

The CRISP-DM data mining life cycle consists of six main phases. The arrows show common dependencies between the different phases. However according to the authors it is always required to go back and forth between phases. Therefore it is not necessary to strictly adhere to the order of the flow (Chapman et al., 1999).

2.6.1 Hierarchical breakdown

The CRISP-DM methodology, as depicted in Figure 2.6, also specifies a hierarchical breakdown of tasks underlying the main phases. This breakdown consists of tasks that are described on four levels of abstraction. The first level of abstraction is known as phases and is shown in the life cycle process model (Figure 2.5).

At the second level generic tasks are described belonging to each phase. These generic tasks are defined in a way to cover all possible data mining cases while keeping in mind technological advancements. See Figure 2.7 for an example overview of generic tasks in the data preparation phase. Furthermore, just as with the flow of events between the main phases of a data mining project, the authors explain that in practice it is often the case that previous actions need to be repeated. Therefore the flow of the tasks should not be considered as a rigid process.

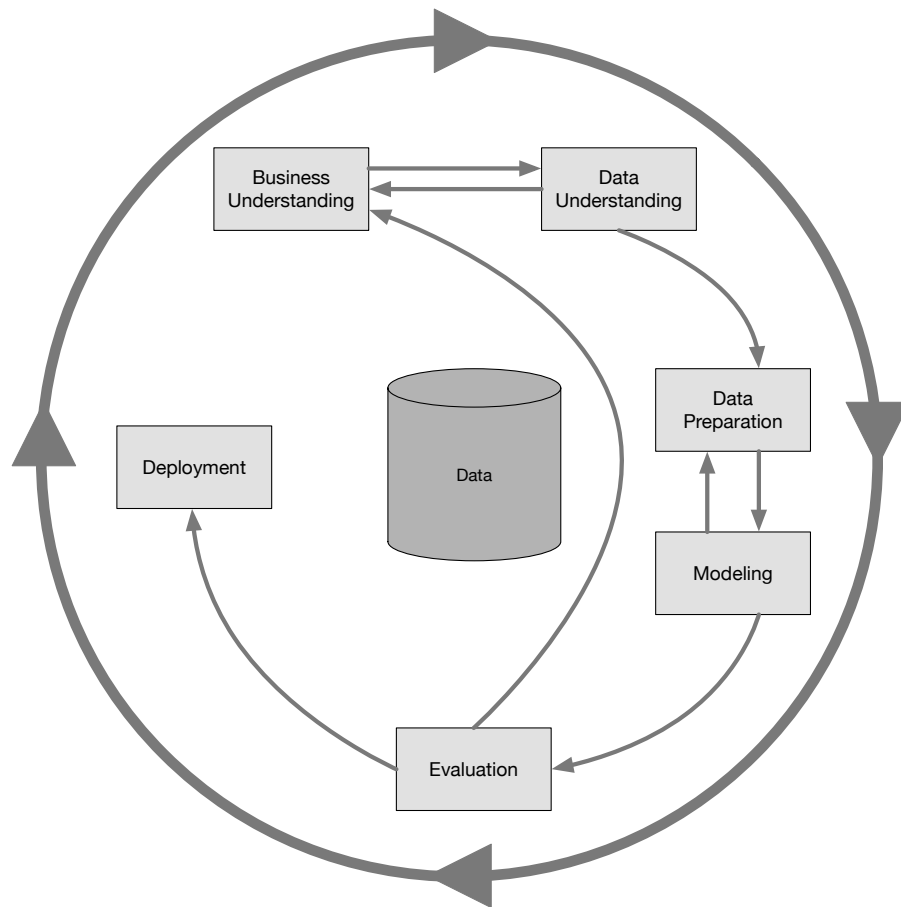


FIGURE 2.5: The CRISP-DM process model lifecycle (Chapman et al., 1999)

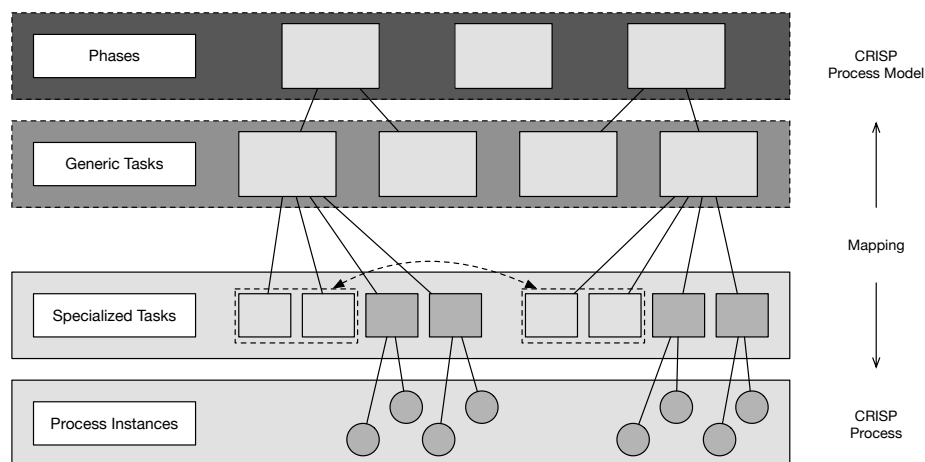


FIGURE 2.6: Four abstraction levels of the CRISP-DM methodology

The third level is called specialized tasks, on this level the generic tasks are specified as decisions on how the task will take shape. For example the generic task called select data which is part of the data preparation phase can be performed by stating that an automatic feature selection algorithm will be applied to select the most fitting attributes.

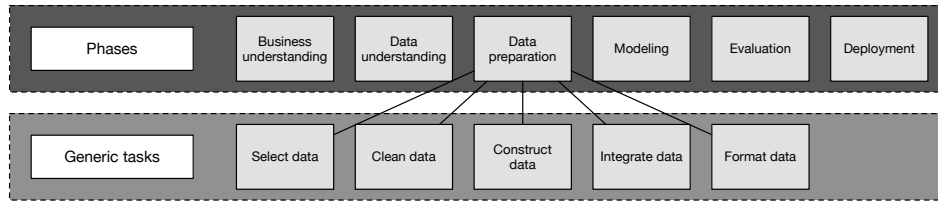


FIGURE 2.7: Generic tasks in the data preparation phase

KDD	Human centered	SEMMA
<ul style="list-style-type: none"> - Linear - Primarily aimed at deriving patterns from data 	<ul style="list-style-type: none"> - Linear - Oriented around a human understandable task overview of DM projects 	<ul style="list-style-type: none"> - Linear - Compact and high level (does not specify concrete tasks)
3PM	EoA	CRISP-DM
<ul style="list-style-type: none"> - Cyclical - Detailed task breakdown for outsourcing DM to a third party - Implementation framework for embedding DM results 	<ul style="list-style-type: none"> - Cyclical - Rigorous pattern definition within each cycle to ensure valid results 	<ul style="list-style-type: none"> - Cyclical - Detailed task breakdown per phase - Concrete to generic task mapping framework

TABLE 2.1: DM process model highlights

Last, the fourth level is called process instances, this layer represents a complete record of an actual data mining project including deviations from the process and / or other particularities (Chapman et al., 1999).

2.7 Process model selection

See Table 2.1 for an overview of the highlights of each process model discussed in this chapter. In this section we will address our selection criteria in relation to the aforementioned models.

The CRISP-DM consortium's collaborative and practical approach towards the creation of their methodology have lead to a wide adoption of the process model. This is supported by repeated poll's on methodology use by KDnuggets.com, a website focussed on data mining, analytics, big data, and data science¹. From 2007 to 2014 CRISP-DM shows to be the top methodology of choice for around 40% of the voters (Piatetsky, 2014). SEMMA and KDD stay below the 10% mark. The poll also specifies an options for custom methods ("my own"), it is noteworthy to see that the number of voters using their own methods has increased over the years. Unfortunately it is not known whether this concerns variants of existing methodologies or new and unique initiatives. See Table 2.2 for an overview of the data.

¹<http://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html>

Data mining methodology usage		
	2007	2014
CRISP-DM	42%	43%
My own	19%	27,5%
SEMMA	13%	8,5%
KDD Process	7,3%	7,5%

TABLE 2.2: Poll on data mining process model usage by KDnuggets.com

Next concerning the level of depth we see that the human-centered approach, the Three-phases model and CRISP-DM contain multiple layers of concepts which allow for a more fine grained mapping of ML techniques to the methodology.

As far as the technical depth of the process models the human-centered approach and the Three-phases model are the only process models to provide concrete information on tools to be used during the different phases. The CRISP-DM refrains from this practice leaving it to the user to define tools and techniques on the third (specialized tasks) layer.

To conclude, compared to the other process models CRISP-DM shows to be the most popular method of choice. Furthermore, the level of depth fits our choice to craft a method based on method fragments while at the same time allowing to map technical ML concepts to each fragment. We will therefore use CRISP-DM and the hierarchical breakdown as the base of our method (fragments).

Chapter 3

Machine learning concepts

In this chapter we present an overview of procedural and technical concepts that are applied in data mining projects. As explained in Chapter 1 we will focus on concepts that are related to solving supervised two-class (binary) classification problems.

The chapter is structured in line with the phases of the CRISP-DM process model. The concepts found are grouped within one of the following phases (sections): data understanding, data preparation and modeling. Since this research project is oriented around the technical side of a DM project non-technical tasks are out of scope. This means that the following CRISP-DM phases will be omitted from this chapter: business understanding, evaluation and deployment.

The techniques outlined in this chapter are applied to the *Iris* data set first used by Sir R.A. Fisher (Fisher, 1936). Until this day it remains a popular data set to teach and demonstrate ML techniques. The data set contains information about the properties (features) of three types of flowers (classes). The features are petal width, petal length, sepal width and sepal length. The set consists of 150 observations (instances), 50 per class.

3.1 Data understanding

Before starting with any data mining project it is important to become familiar with the data that will be analyzed. The goal is to improve one's understanding of the data by using statistical tools to summarize, plot and review datapoints in the data set. This practice is called exploratory data analysis (EDA) (Brownlee, 2014b).

The term EDA was coined by Tukey (1977), he advocated the use of statistical tools to explore data as a means to generate hypotheses through a better understanding of the data. This is considered an alternative approach to confirmatory data analysis where a full understanding of the data is assumed beforehand. However, Brownlee (2014b) explains that this is rarely the case in data mining projects.

In this section we will show three EDA techniques that provide insight into the properties and relations between variables (features) in a data set. We will be looking at histogram graphs, pairwise scatterplots and correlation plots.

The visualizations will be created by using the Seaborn graphics library which is an extension to the Matplotlib scientific graphics package (Hunter, 2007). Seaborn provides frameworks to draw common

statistical visualizations and integrates with other data science tools in the Python data science stack (Waskom et al., 2015).

3.1.1 Histogram graphs & Pairwise scatterplots

The Seaborn graphics library provides a method to generate an overview where histogram graphs and pairwise scatterplots are combined in one image, it is called the pairplot. See Figure 3.1 for a pairplot of the Iris data set.

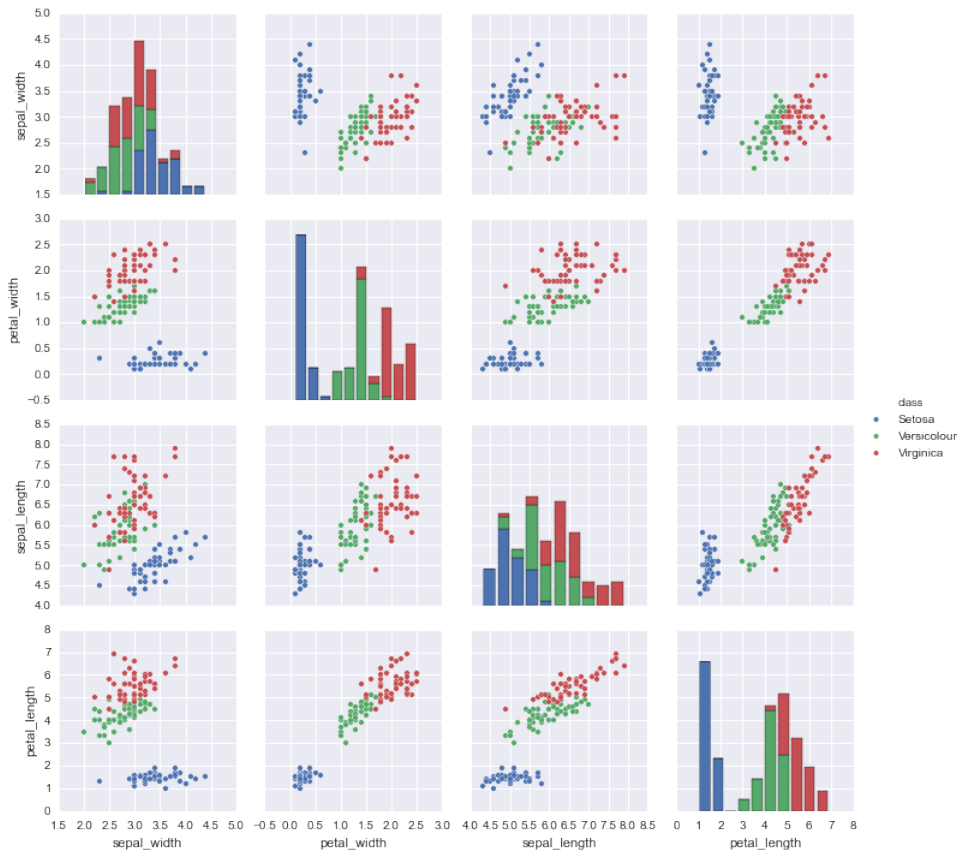


FIGURE 3.1: Histograms and pairwise scatterplots based on the Iris data set

When exploring the data from this perspective the key is to look for areas where classes are (more or less) separated from each other. For example, the histogram graphs show that the features petal width and petal length have minimal overlap between the different classes. Thus they can be considered as informative features (Metzen, 2015). Next the scatterplots show that the Setosa class (blue dots) of flowers remains completely (linearly) separated from the other classes. This means that the Setosa class can be clearly separated from the other classes when using ML algorithms.

3.1.2 Correlation plot

When moving on to the correlation plot as depicted in Figure 3.2 we see that our observations are confirmed. The features petal length, petal width and sepal length are strongly correlated with the class which means that they

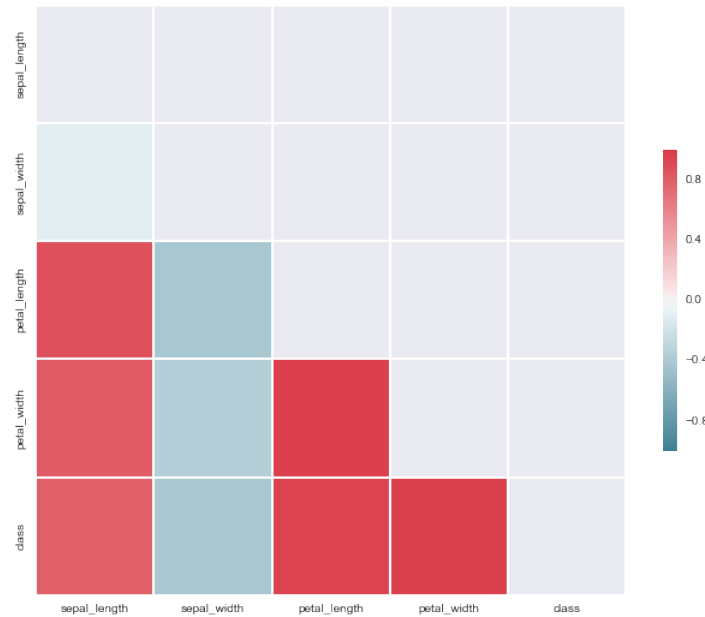


FIGURE 3.2: Correlation plot based on the Iris data set

are informative. Furthermore we observe that the features are also highly correlated with each other. Which means that we have redundant features in the data set. It is important to keep in mind that some classification techniques like e.g. naive bayes do not work well with redundant features. In this particular case the importance of the redundant features will be over inflated by the naive bayes classifier.

3.2 Data preparation

In this section we address data preparation techniques that optimize data for use with ML algorithms. Lan, Frank, and Hall (2011) refer to this practice as data engineering. We discuss feature selection, construction, transformation and their possible benefits in relation to specific algorithm classes.

3.2.1 Feature selection

Not all features in a given data set have the same informative importance or any importance at all. Also some data sets can be *high dimensional*, which means that they have a large number of features. This can be problematic as some classification algorithms are designed to make the most of the data that is presented to them. In these cases even irrelevant features will eventually be included in the model. In other words the model will be *over-fitted* to the data which means that the classification algorithm has included the noise as an integral part of the model (Tang, Alelyani, and Liu, 2014). Classification algorithms like decision trees, rule based algorithms, linear regression, instance-based learners and clustering algorithms are known to deliver lowered performance when they are presented with irrelevant features. This does not apply to the naive bayes classification algorithm, it remains robust when presented with irrelevant data. However as explained

in subsection 3.1.2 redundant features should be removed (Lan, Frank, and Hall, 2011).

The solution is to select the subset of features that are informative and discard the features that are irrelevant (noise). Besides manual selection two types of automated strategies exist to accomplish this task. The first is called the *filter* approach, in this approach each feature is individually scored and the top ranking features are selected for use. The approach can use scoring metrics like information gain, chi squared and the correlation coefficient. The second approach is known as *wrapper*. Instead of individual features candidate sets of features are evaluated by training a model. The *forward selection* search strategy starts with an empty candidate set. Features are added one by one as long the performance of the model is improved. The *backward elimination* search strategy works the other way around by eliminating features until the performance of the model stops showing improvement (Guyon and Elisseeff, 2003; Flach, 2012).

3.2.2 Feature construction

Constructing new features from the original features can help optimize the learning task of a classification algorithm. There are two general approaches to feature construction: (automated) feature extraction and (manual) feature engineering.

Feature extraction

Although not clearly defined in literature, feature extraction is mostly referred to as a method to derive a new set of features by applying a *projection* to the data. Lan, Frank, and Hall (2011) explain the concept of projecting as "a kind of function or mapping that transforms data in some way". Projections serve the goal of deriving highly informative and non-redundant features that support the learning task. The practice is related to the concept of *dimensionality reduction* since the number of informative features that are derived is limited to a low number.

A frequently cited projection method is principal component analysis (PCA). In simplified terms the method exposes the properties of a data set which allow to summarize it as best as possible (Amoeba, 2015). Without delving too deep in the mathematical background of PCA, the method determines which combination of features explain the maximum amount of variance in the data and aggregates these findings as the principal components. Each principal component represents an axis (dimension) once applied to transform the original data set. Typically the first two or three components explain the majority of variance of a data set. This enables visualizing a representation of the original data set along 2 or 3 axes, depending on how many principal components are chosen by the user. Furthermore, this representation emphasizes properties where the instances are very different from each other which means that opportunities for separation between classes may become more clearly visible (Wold, Esbensen, and Geladi, 1987; Raschka, 2014).

See Figure 3.3 for an example where PCA is applied to the Iris data set. The data set originally has 4 dimensions which have been reduced to 2 dimensions. This allows us to visualize the data along 2 axes. We observe

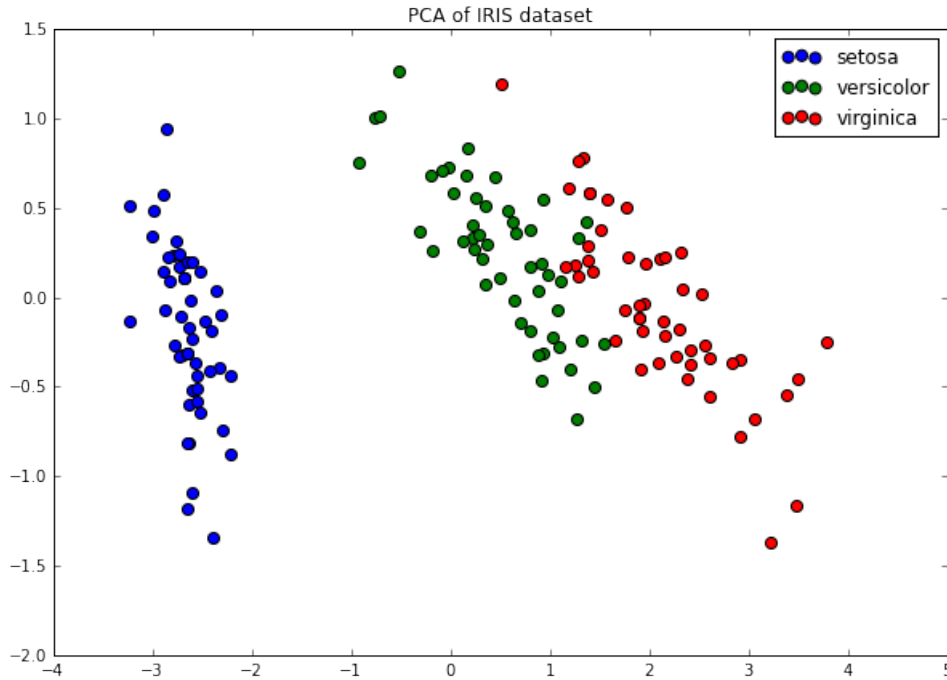


FIGURE 3.3: Principal component analysis applied to Iris data set

an overview that is similar to some of the individual feature pairplots in subsection 3.1.1, however this visualization represents all the features of the original data set.

Feature engineering

Not all features immediately convey their informative value. Further processing may be required to expose a different angle on the data set. A simple example of this is calculating the age of participants when provided with a data set that originally only holds the date of birth of each subject. Another more complex example, as we will demonstrate in the case study of this thesis, is the calculation of the number of unique places that have been visited by a participant based on raw GPS data. This way of constructing features requires domain knowledge, in other words one should know what to look for. It is therefore difficult to establish a pattern that can be modelled. However, it remains a vital activity in the process of creating informative data sets (Locklin, 2014).

3.2.3 Feature transformation

Transformations are applied to improve the interpretative potential of data sets. This is partially dependent on the type of classification algorithm that will be used. We will discuss recommended datatypes in section 3.3 where model types are addressed.

Continuing with our example on ages of participants. With domain knowledge on the subject it may become clear that participants between ages 20 - 30 are of special interest. Therefore another binary feature may be

engineered *is_between_20_and_30* which would hold the value 1 for a participant aged 25. This decomposition to binary values helps to get more out of simpler linear models and decision trees (Brownlee, 2014a).

Transformations are possible in different directions depending on the current and required type of data. See Table 3.1 for an overview of transformation types outlined by Flach (2012).

▼ to, from ►	<i>Quantitative</i>	<i>Ordinal</i>	<i>Categorical</i>	<i>Boolean</i>
<i>Quantitative</i>	normalization	calibration	calibration	calibration
<i>Ordinal</i>	discretization	ordering	ordering	ordering
<i>Categorical</i>	discretization	unordering	grouping	
<i>Boolean</i>	thresholding	thresholding	binarization	

TABLE 3.1: Datatype transformation directions by Flach (2012)

According to the author both *binarization* and *unordering* can be clearly deducted from the source data. We explained the notion of binarization in the introduction of this subsection, unordering simply discards the ordering of the feature values meaning that the values are considered as categorical. We will continue to address transformation operations that require additional decision making during their application.

Tresholding and discretization

Tresholding is the process where a quantitative or ordinal feature is transformed to a boolean feature. Discretization transforms quantitative features to ordinal or categorical features called *bins*. See Table 3.2 for an example of these transformations.

Age	Tresholding (manual split on 30)	Discretization (manual boundaries)
6	0	0-10
17	0	11-20
13	0	21-30
29	0	31-40
36	1	51-60
31	1	
24	0	
54	1	
59	1	
16	0	

TABLE 3.2: Example of tresholding and discretization

Both transformation techniques face the same challenge, how to accurately determine the optimal splitting values and boundaries of the bins. The manual way as demonstrated in Table 3.2 carries the risk of losing fine distinctions in the data where instances are unevenly divided per split / bin (Lan, Frank, and Hall, 2011). Fortunately, supervised and unsupervised methods exist that support these transformations. Unsupervised methods adhere to the principles of evenly dividing instances over bins resulting in intervals of different sizes (*equal-frequency binning*). Or in the case of tresholding summary statistics like the mean and the median often are sensible

points of splitting the data. Supervised methods incorporate the class data in scoring methods to find more refined boundaries and splitting points. Dougherty, Kohavi, and Sahami (1995) found that supervised discretization methods enable improved classification results over unsupervised methods.

Normalization and calibration

Flach (2012) explains that thresholding and discretization are transformation methods that remove the scale from quantitative features. Normalization and calibration work in the other direction by adapting and / or adding scale information to features.

Normalization converts the values to of a quantitative feature to Z-scores, in other words the number of standard deviations a value is separated from the median. Normalization ensures that different quantitative features measured on different scales do not negatively influence the learning task.

Calibration adds scale information to ordinal and categorical features. The method uses class information (supervised) to ensure meaningful weighting of the calibrated values. Calibration allows to use classifiers that can only handle scaled data in combination with categorical and ordinal features. Furthermore by using calibrated data this leaves the option to the classifier to choose on how to best treat a feature (categorical, ordinal or quantitative).

Weight	Diabetes	Calibrated weight	Weight	Diabetes	Calibrated weight
130	Y	0,83	81	N	0,43
127	Y	0,83	80	Y	0,43
111	Y	0,83	79	N	0,43
106	Y	0,83	77	Y	0,43
103	N	0,60	73	N	0,40
96	Y	0,60	68	N	0,40
90	Y	0,60	67	Y	0,40
86	N	0,50	64	N	0,20
85	Y	0,50	61	N	0,20
82	N	0,43	56	N	0,20

TABLE 3.3: Normalization example, adapted from Flach (2012)

3.3 Modeling

During the modeling phase of a DM project one or more algorithms are selected. Next, the algorithms are applied to the prepared data set. The resulting models are then evaluated using e.g. quality measures and error rates (Chapman et al., 1999). As explained in section 1.4 algorithms require configuration of parameters. Finding the optimal parameter settings is essential in attaining robust and accurate models. This makes the aforementioned process cyclical in nature. After tweaking the parameter settings the modeling and evaluation tasks are repeated until they yield satisfactory results.

In this section we will be looking at model types including popular algorithm implementations meant for classification. We will outline the general properties of each model type and provide (illustrative) examples where necessary. Next we will look at various strategies to find optimal parameters for algorithm implementations. We conclude this section with an overview of model evaluation techniques and explain how to interpret the results.

3.3.1 Model types

The model types discussed in this subsection represent a majority sample of all the available types. The types are chosen because they are frequently referenced in literature and cover a wide area of applications. We feel that these concepts will provide an adequate starting point to explore modeling using ML techniques.

Tree models

Tree models, also known as decision trees, are a popular method of choice for modeling data. Decision trees are easy to understand and help to uncover the most informative / important features of a data set (Flach, 2012).

Tree modeling algorithms build decision trees by ranking features on how well they divide the data set. The first node, or the root node of a tree is always represented by the feature that best divides a data set. Next the method determines where the next best splitting points are and encodes these values as decision points on how to traverse down the tree (Myles et al., 2004).

Decision trees are prone to *overfit* the training data meaning that properties that cannot be generalized to the real world are included as part of the tree model. Kotsiantis, Zaharakis, and Pintelas (2007) explains two methods to deal with the problem over overfitting decision trees. The first method encompasses preventing the learning algorithm from achieving a perfect fit on the training data. This is undesirable since a perfect fit means that non-informative features are included in the model as well. To achieve this stopping criteria will have to be defined for the decision tree inducing algorithm. Fortunately, most decision tree inducing algorithms have *parametrized* values like the maximum number of nodes and leaves. The second method to prevent overfitting is called *pruning* and refers to the act of applying algorithms to intelligently remove leaves and nodes while keeping an eye on the performance of the model. A good rule of thumb is that a tree with fewer leaves is preferred because it is less likely to overfit the data.

Tree models are by their nature designed to work with categorical data (Kuhn and Johnson, 2013). However well known implementations, like C4.5 (Quinlan, 2014) have overcome this limitation and can deal with both categorical and numeric features. Furthermore, tree models do not require the data set to be normalized in order to gain optimal results. As an example we applied a decision tree algorithm to the Iris data set, see Figure 3.4 for the graphical overview.

Figure 3.4 clearly shows the linearly separable setosa class of which all samples are found to have a petal length shorter than 2.45 centimeters.

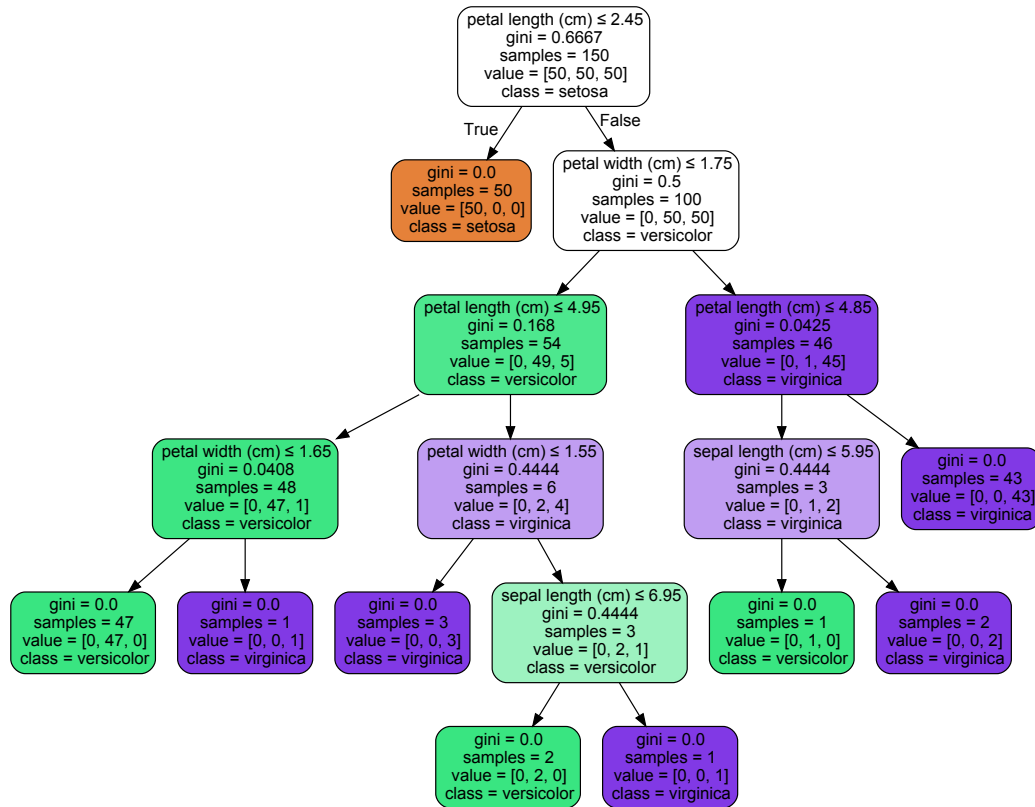


FIGURE 3.4: Decision tree model of the Iris data set

The third and fourth layers of the tree show how the majority of the versicolor and virginica class are separated based on the features petal width (≤ 1.65) and petal length (≤ 4.85). Observe how the fifth and sixth layers classify the remaining few datapoints while adding a substantial amount of complexity to the tree. These branches could be candidates for pruning to prevent overfitting.

Linear models

Linear models approach the learning task from a geometric perspective reasoning about data as points in a (Cartesian) coordinate system. A practice very similar to the principles outlined in section 3.1 where we explain what to look for when exploring the data. Modeling techniques apply geometric concepts such as lines and planes (2-D surface) to structure points in coordinate spaces and in turn enable classification of these points (Flach, 2012). We discuss two algorithm implementations of the linear model type known for their high accuracy on classification problems.

Multi layer perceptron with backpropagation (MLP): MLP is an algorithm implementation that belongs to the class of *artificial neural networks* (ANN). At their core neural networks establish linear relationships between numerical / quantitative features including the strength of these relationships. These are recorded in hidden layers that are part of the model. The process of learning is performed by ANN algorithms which tune the strength values.

The main difference between the single layer and multi layer perceptron is that the latter is able to deal with data sets where the classes are not linearly separated in geometric space. MLP works around this with multiple hidden layers that allow to closely define where classes are expected to be separate from each other. The learning task initiated by the backpropagation algorithm. The algorithm repeatedly runs input data through the model with the goal of measuring the error rates. This information is fed back to the ANN algorithm which in turn tunes the strength of the relationships between the nodes in each hidden layer. This process is repeated until the error rates fall within acceptable bounds (Karayiannis and Venetianopoulos, 2013). See Figure 3.5 for an illustration of the layers in an MLP.

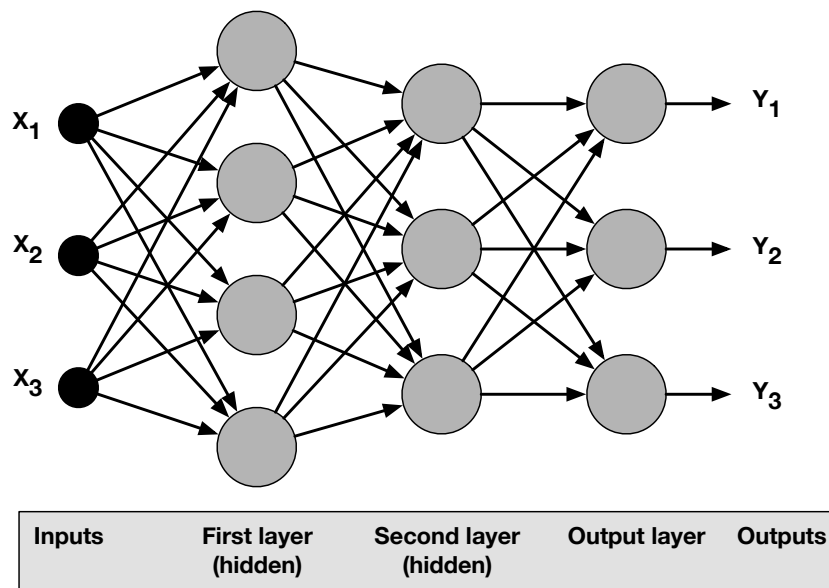


FIGURE 3.5: Overview of MLP layers

Support Vector Machines (SVM)

The SVM classification strategy entails finding an opportunity for linear separation between classes in geometric space, more specifically an SVM determines the optimal placement of the line (decision boundary) that separates two classes. The datapoints close to this line are used to determine the orientation and position of the line and are called *support vectors*. The goal is maximize the *margin* which is defined as the distance between the decision boundary and the nearest support vectors (Murty and Devi, 2011).

In cases where classes are not linearly separable SVM's can be adapted to use a *soft margin*. Since misclassification is inevitable in trying to separate non linearly separable classes, the goal of the classification algorithm is to keep the error rate as low as possible while still maximizing the margin (Cortes and Vapnik, 1995).

Both SVM's and ANN's are primed towards handling numeric (quantitative) data, however this does not exclude analysis of categorical data. A tried-and-true method to use categorical data is to binarize the values which means that each categorical value is processed as a separate attribute which is marked as either true (1) or false (0) (Hsu, Chang, and Lin, 2003; Blattberg, 2003), see Figure 3.7 for an example.

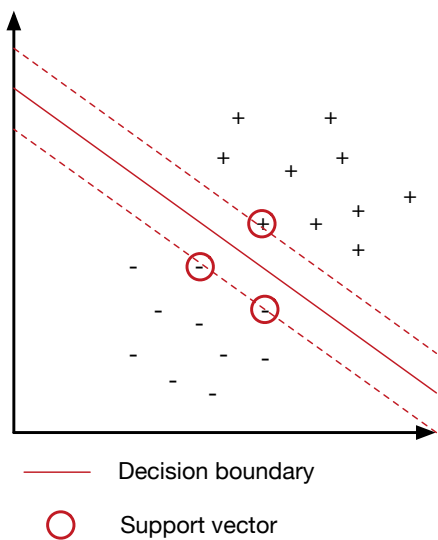


FIGURE 3.6: SVM decision boundary placement, adapted from (Flach, 2012)

Name	Sex		Name	Female	Male
John	Male	> Binarization of categorical data	John	0	1
James	Male		James	0	1
Jenny	Female		Jenny	1	0

FIGURE 3.7: Representation of binarized categorical data

SVM’s are scale sensitive meaning that unnormalized data may distort the outcomes of the model, ANN’s do not suffer from this problem but it is found that normalized data helps the algorithm to converge on a solution in a shorter timespan.

Probalistic models

Probalistic classification models determine class membership as a function of the probabilities that specific feature values belong to a certain class (Flach, 2012), this is assesed by applying Bayes’ theorem. In short Bayes’ theorem describes how to determine the probability of an event given the conditions related to that event (Bolstad, 2013). This particular perspective on machine learning revolves around the reduction of uncertainty. Modeling initiates with maximum uncertainty about the prior probability of an instance belonging to a certain class. The act of learning is performed by adjusting the probability estimates after screening each instance and its class information, thus leading to reduced uncertainty about class membership.

The **naive bayes** modeling technique follows the principles as explained above. The technique is called naive or simple because it assumes that all features are independent (non-correlated) of each other. Although complete feature independence is never true in reality the modeling technique is known for its accuracy in real world situations (Domingos and Pazzani, 1997).

It is found that the modeling technique works best when provided with discretized (categorical) data (Yang and Webb, 2003). Basic implementations of the technique do not require additional parameter settings, however depending on the distribution of the data different variants of the modeling technique may be better suited for the task e.g. gaussian naive Bayes, multinomial naive Bayes and Bernoulli naive Bayes (Pedregosa et al., 2011). Last the model does not require the normalization preprocessing step since this is performed by design.

Rule models

Rule models are very similar to decision trees in the sense that each branch represents a rule that splits the data. However rules can overlap whereas branches cannot, which potentially leads to a loss of information.

A typical rule modelling technique works by generating a rule and removing the samples that fall under that rule. This is known as the **covering algorithm** (Michalski, 1975). The approach is repeated until the stopping criteria are met, these criteria are in place to prevent noise from being modelled as rules. This is similar to how tree models can be restricted in their growth to avoid overfitting.

Although not a supervised technique it is noteworthy to mention the concept of *association rule mining* (ARM) by means of the **apriori algorithm**. Association rule mining aims to uncover interesting relationships between all the variables of a data set instead of focussing on the class. A frequently cited example of ARM is market basket analysis where the technique is used to study purchasing patterns of customers e.g. to see which products are often bought together (Chen et al., 2005).

Petalwidth	Class
1.75-inf	Iris-virginica
0.8-1.75	Iris-versicolor
-inf-0.8	Iris-setosa

TABLE 3.4: Rule model applied to the Iris data set

See Table 3.4 for an example of a rule model that has been applied to the Iris data set. We observe that the algorithm determined that only three rules related to one feature were necessary to correctly classify 92% of the data set.

Ensembles

Model ensembles are known for consistently delivering high performance in terms of classification accuracy. Ensemble methods generally work by constructing multiple models from the data set which may be resampled or reweighted. The predictions of these models are then combined either by voting or averaging the results. These methods reduce the effect of random fluctuations in single models which negatively affects classification accuracy (Flach, 2012).

Bagging is an ensemble technique where different models are created based on random samples of the data set. In this technique samples are taken *with replacement* which means that they can occur multiple times in

one sample. Because of this even if the sample size is as large as the original data set they are bound to become different from each other. This is of particular importance because it enables variations between the models that are created. Predictions can be gained through voting or averaging. With voting the prediction of the majority of the models wins. Averaging can be used when models generate numerical values. The technique is found to be useful in combination with tree models since trees are sensitive to variations in data sets. The combination of bagging with tree models is also known as the **random forest ensemble** (Breiman, 1996).

Boosting is slightly similar to bagging but has a different approach towards training models. The boosting process starts with training a classifier on a data set. Misclassified instances are reweighted and duplicated to data sets that are used to train the next model. With each iteration models will be forced to focus on instances that are difficult to classify. Predictions are gained through a weighted majority vote by all the models. A well known boosting algorithm is **AdaBoost**, it can be used for classification and regression (Freund and Schapire, 1995).

Both random forests and adaboost are based on tree models and therefore do not require the data to be normalized. Both can handle quantitative and categorical data.

3.3.2 Parameter optimization

Recall from section 1.4 that the performance of algorithms is dependent on how they are configured, a problem known as (*hyper*) *parameter optimization*. Getting optimal performance from a modeling technique means finding the right (combination of) parameter settings. The optimal settings will be different for each data set which necessitates an automated means of determining the optimal values. Currently there are three strategies to deal with the parameter optimization problem, these are outlined below.

Grid search

This search strategy essentially revolves around repeating the process of training and evaluating models based on a predefined grid of parameters. Grid search is exhaustive which means that every possible combination of parameter values will be evaluated. See Table 3.5 for an example of a searchgrid definition. The first row shows three possible values to be tested for the max features parameter. The second row has two features for the criterion parameter. This means that the model will be trained and evaluated for a total number of six times.

Parameters	Values
max_features	[1,3,10]
criterion	["gini","entropy"]

TABLE 3.5: Searchgrid example

Popular ML toolkits like Scikit-learn provide convenience methods that help in preparing, running and evaluating the search (Pedregosa et al.,

LISTING 3.1: Gridsearch output

```
1 GridSearchCV took 0.99 seconds for 6 candidate parameter settings.
2 Model with rank: 1
3 Mean validation score: 0.927 (std: 0.009)
4 Parameters: {'max_features': 10, 'criterion': 'gini'}
5
6 Model with rank: 2
7 Mean validation score: 0.924 (std: 0.009)
8 Parameters: {'max_features': 10, 'criterion': 'entropy'}
9
10 Model with rank: 3
11 Mean validation score: 0.922 (std: 0.005)
12 Parameters: {'max_features': 3, 'criterion': 'gini'}
```

LISTING 3.2: Randomsearch output

```
1 RandomizedSearchCV took 0.27 seconds for 2 candidates parameter settings.
2 Model with rank: 1
3 Mean validation score: 0.925 (std: 0.011)
4 Parameters: {'max_features': 3, 'criterion': 'gini'}
5
6 Model with rank: 2
7 Mean validation score: 0.916 (std: 0.005)
8 Parameters: {'max_features': 3, 'criterion': 'entropy'}
```

2011). We applied the searchgrid in combination with a random forest classifier to the Iris data set. See listing 3.1 for an example of the evaluation results.

Last, it should be noted that, due to its exhaustive nature, gridsearch is a costly strategy in terms of processing power and time. When these resources are limited random search may be an acceptable alternative.

Random search

Random search is very similar to gridsearch but aims to solve the problem of heavy computation costs. This search strategy chooses a random set of parameter values, furthermore the number of evaluations can also be limited which give precise control over the runtime of a search. Despite the random nature of this approach it is known to perform equally well and sometimes even better than grid search at reduced computational cost (Bergstra and Bengio, 2012). See Listing 3.2 for an example of the random search strategy. The search was limited to 2 tries out of 6 possibilities.

Bayesian optimization

The third and last search strategy falls in between grid search and random search. With each evaluation run bayesian optimization employs intelligent methods to estimate the next best set of parameters, incorporating the information found during previous runs. Snoek, Larochelle, and Adams (2012) explain that this process is computationally heavy but brings the benefit of finding optimal parameter sets in a limited number of runs. This makes

bayesian optimization a good choice for optimizing search problems with a high dimensional parameter space.

Auto-WEKA is an experimental machine learning toolkit that almost completely relies on bayesian optimization techniques to generate models (Thornton et al., 2013). The toolkit is unique in the sense that it considers the choice for the modeling technique as part of the problem space as well. This relieves potential users from having to manually select and test algorithms, instead Auto-WEKA uses all the algorithms that are part of the WEKA toolkit and determines which algorithm generates the best results for a given data set.

3.4 Model evaluation

The model evaluation task is the final step in the modelling phase of the CRISP-DM process model. However for hierarchical clarity we decided to address this task in a separate section. The evaluation task is performed to determine the generalizability of the models that are created during the modeling phase. We can do this for multiple modeling techniques to determine which algorithm works best for a given data set. We start by describing various performance measures that are relevant to binary classification problems. Furthermore we will also touch upon resampling methods since they are necessary to determine accurate performance metrics.

3.4.1 Performance measures

Confusion matrix based measures

A confusion matrix is a tabular representation of the classification results of a model. It lists the classes of a data set in relation to the classification results. See Table 3.6 for an example adapted from Markham (2014).

n=21	Predicted:		
	NO	YES	
Actual: NO	TN = 13	FP = 4	17
Actual: YES	FN = 2	TP = 2	4
	15	6	

TABLE 3.6: Confusion matrix example, adapted from Markham (2014)

The example shows the analysis outcomes of a two class (binary) data set, e.g. whether a participant is classified as a patient or not. The data set contains information about 21 participants from which 4 participants are patients and 17 participants are not. The analysis outcomes show that from these groups 6 participants are classified as patients and 15 participants are classified as non-patients.

The abbreviations TP and TN stand for true positives and true negatives, this refers to correct classification decisions. In this case this means

that participants are correctly classified as either a patient or non-patient. FP (false positives) and FN (false negatives) stand for incorrect classification decisions. A false positive means that a participant is classified as a patient while this participant is a non-patient, this is also known as a *Type I error*. The other way around, a false negative, means that a participant is classified as a non-patient while this participant really is a patient (*Type II error*).

This allows us to calculate the following performance measures from the confusion matrix:

- **Accuracy:** measures how often a classifier is correct:
 $(TP+TN)/total = (2+13)/21 = 0,71$
- **Error rate:** measures how often a classifier is wrong (also known as the misclassification rate):
 $(FP+FN)/total = (4+2)/21 = 0,29$
- **True positive rate:** measures how often a classifier correctly classifies as yes (also known as *sensitivity*):
 $TP/actual\ yes = 2/4 = 0,5$
- **False positive rate:** measures how often a classifier incorrectly classifies as yes when it's actually no:
 $FP/actual\ no = 4/17 = 0,23$
- **Specificity:** measures how often a classifier correctly classifies as no:
 $TN/actual\ no = 13/17 = 0,76$
- **Precision:** measures how often a classifier is correct when it predicts yes (not the same as true positive rate):
 $TP/predicted\ yes = 2/6 = 0,33$

Our adapted example shows that although our accuracy of 0,71 may initially seem good, the true positive rate of 0,5 in effect means that correctly predicting whether a participant is a patient or not essentially comes down to random guessing. Misrepresentation by the accuracy measure is a common problem where classes are imbalanced in data sets. We will therefore look at a measure that does not suffer from this problem, the *receiver operating characteristic* (ROC) (Japkowicz and Shah, 2015).

ROC (AUC)

In machine learning context the ROC represents a plot of the true positive rate (TPR) against the false positive rate (FPR) over all possible decision thresholds, it applies only to binary classification problems. Most ML algorithms assign a score to an instance e.g. 0.8 which is thresholded to either 0 (no) or 1 (yes). The decision threshold usually is 0.5 which means that our example value 0.8 would be thresholded to 1. ROC values are calculated by determining the TPR and FPR values for each possible threshold between 0.0 and 1.0 (Hallinan, 2014).

See Figure 3.8 for an example plot of an ROC. If the classifier performs well at separating both classes the TPR will remain high before curving off

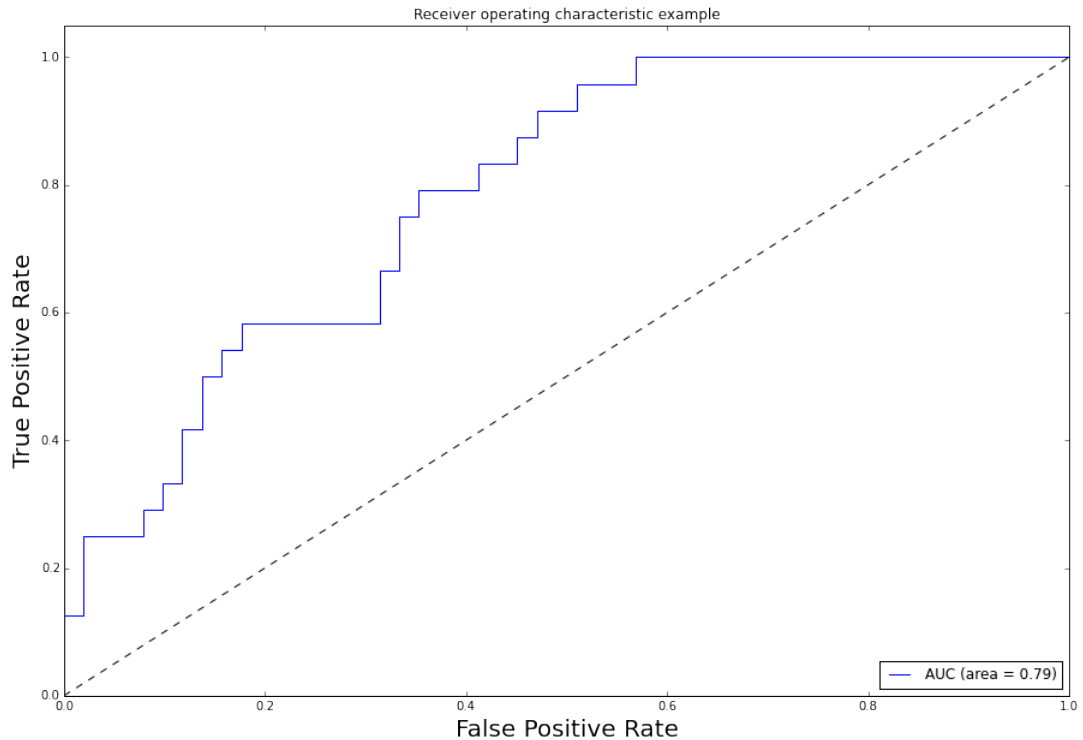


FIGURE 3.8: ROC example based on the Iris data set

towards higher FPR values. In other words, the farther the curve is removed from the diagonal dotted line the better a classification model performs. This outer left corner is also known as *ROC heaven* in the ROC space (Mladenic, 2003).

The ROC allows to derive a metric known as the *area under the curve* (AUC) which represents the proportion of the box that resides under the line. The higher this proportion the better a classifier performs, values between 0.80 and 0.90 are found to be acceptable while 0.90 and higher values are good. Furthermore, keeping in mind the medical context of this research project, since the AUC is built on components like sensitivity and specificity it is also found to be an effective method to validate diagnostic tests (Kumar and Indrayan, 2011).

3.4.2 Resampling methods

Resampling is applied to data sets to support the process of testing the predictive capabilities of an algorithm. We discuss frequently cited resampling techniques applicable to different situations below.

Resubstitution and holdout

Resubstitution means that models are trained and tested on the same data set. This is considered as a bad practice in ML since overfitting the data will not be detected in this way. To determine the generalizability of a model it should be tested on unseen instances. This is where the holdout- and subsequent methods come in.

The holdout method entails splitting the data set in a training and testing set, e.g. according to a 70% - 30% ratio. The model is first trained using the training set, afterwards it is tested on the unseen instances of the test set. The holdout method works well in cases where the data set is large and representative of the problem being analyzed (Japkowicz and Shah, 2015).

Stratified k-fold cross validation

In k-fold cross validation the data set is split into k equal parts. The data is then trained on k-1 parts and then tested on the remaining part. This process is repeated until all the different parts are used for evaluation. So for example, in 10-fold cross validation the training and testing phase will be repeated for a total of ten times. This process is known to work well for smaller to medium sized data sets and generally leads to low error rates (Japkowicz and Shah, 2015). The term stratified refers to the process to counter class imbalance problems by evenly dividing classes over the k number of folds.

Leave one out

The leave one out method is similar to cross validation except that each instance represents its own part. So for example, in a data set containing 30 instances the set is trained and evaluated 30 times using a training set of 29 instances and a test set of 1 instance. This method is applied when a data set is not large enough to split in a meaningful number of k folds.

Bootstrapping

Bootstrapping is an alternative to k-fold cross validation and also suitable to make the most of smaller data sets because it dampens the variance of the results. This resampling method works by drawing a random sample with replacement from a data set. The term with replacement means that the same instance can be included multiple times in one sample. At each run the sample represents 63,2% of the data set, the training set consists of all the instances that have not been selected for the sample. The process is typically run for at least 200 times or more to achieve accurate results. The result itself is the average of the performance metric (e.g. accuracy) over all the runs.

3.5 Accuracy versus transparency

The lack of research and general talk on the topic of model transparency is illustrative of a field that is strongly focussed on improving accuracy related challenges. While there is nothing wrong with the pursuit of this goal, on occasion the need for transparent models also surfaces in the body of literature. In particular when it concerns *decision support systems* where it must be clear how a system came to a certain decision / classification (Johansson, Niklasson, and König, 2004; Olson, Delen, and Meng, 2012; Kamwa, Samantaray, and Joós, 2012b; Allahyari and Lavesson, 2011).

There is consensus in the literature about the types of algorithms that are known to yield transparent and non-transparent (*black box*) models. Both

tree and rule models are considered as transparent and highly interpretable. On the other hand artificial neural networks, support vector machines and ensembles like the random forest are considered as black boxes (Johansson, Niklasson, and König, 2004; Olson, Delen, and Meng, 2012; Kamwa, Samantaray, and Joós, 2012b).

Currently there is no common ground on the subject of tree and rule model complexity. Johansson, Niklasson, and König (2004) note that the interpretative value of a "bushy", in other words complex, decision tree should be questioned. The same should apply to complex rule models. However in a, first of its kind, study on model understandability Allahyari and Lavesson (2011) found evidence that the assumption where simpler models are considered as more understandable does not always hold as true.

The choice between a transparent and non-transparent modeling technique is not immediately obvious since, as explained in section 1.4 there is a tradeoff to be had between accuracy and transparency. Kamwa, Samantaray, and Joós (2012b) found that black box modeling techniques have better classification / prediction performance and that the tradeoff with better interpretable solutions is unavoidable. We found two solutions in the body of literature that aim to bridge this gap.

First, Johansson, Niklasson, and König (2004), Martens et al. (2007), and Setiono (2003) propose to extract comprehensible information in the form of rules and trees from black box modeling techniques like ANN's and SVM's. The practice delivers comprehensible information but is criticized for being unrepresentative of the original model due to oversimplification (Cortez and Embrechts, 2013).

Second, Lou, Caruana, and Gehrke (2012) approach the problem from the opposite direction by improving the performance of a transparent modeling technique to a level where it competes with its black box counterparts. They use a variant of linear modeling known as *generalized additive modeling* (GAM) enriched with information on pairwise interactions between features (Lou et al., 2013). This allows to retain the explanatory value of linear models and at the same time achieve high classification accuracy. The technique exposes the contribution of each feature in relation to the outcome values. It has been applied to analyze medical data sets on the subjects of pneumonia risk and hospital readmission within 30 days. In this study the authors state to have achieved a high level of accuracy. Furthermore the risk scores generated for each feature enabled the discovery of previously unseen patterns (Caruana et al., 2015).

Chapter 4

Method fragments

In this chapter we will tie the concepts as discussed in Chapter 3 together and present the method fragments used to perform supervised binary classification tasks. The method fragments are based on the base structure of the CRISP-DM process model as discussed in Chapter 2. Furthermore, our modelling approach of choice will be the method engineering technique as proposed by Weerd and Brinkkemper (2008) adopted from Saeki (2003).

The technique is used to visualize relations between activities (the process) and the corresponding deliverables (concepts). The resulting visualization is called a *process-deliverable-diagram* (PDD) and consists of a conjoined UML activity diagram and an UML class diagram. The activity diagram is placed on the left-hand side and shows the process. The class diagram is placed on the right-hand side and displays the deliverables of the process. The method fragments are accompanied by tables explaining the activities and concepts depicted in the diagrams (Weerd and Brinkkemper, 2008).

In section 4.1 we present the high level domain independent method fragments that form the boilerplate for a data mining project focussed on using machine learning techniques. Section 4.2 zooms in on a specific data preparation step as applied in the Behapp project. It concerns the conversion of raw location data to an informative feature.

4.1 General method fragments

The general method fragments as shown in this section are concerned with the main data processing phases of the CRISP-DM process model. The actions described in the data preparation and modeling & evaluation method fragments can be piped together leading to a continuous flow between the fragments.

4.1.1 Data understanding

The data understanding phase as depicted in Figure 4.1 revolves around generating visualizations and tables to gain a first insight into the properties of the data set and the relationships between the features. A high number of features makes these deliverables difficult to interpret. Therefore the activity flow shows that in cases of high dimensional data sets it is recommended to pre select a set of features using a ML algorithm.

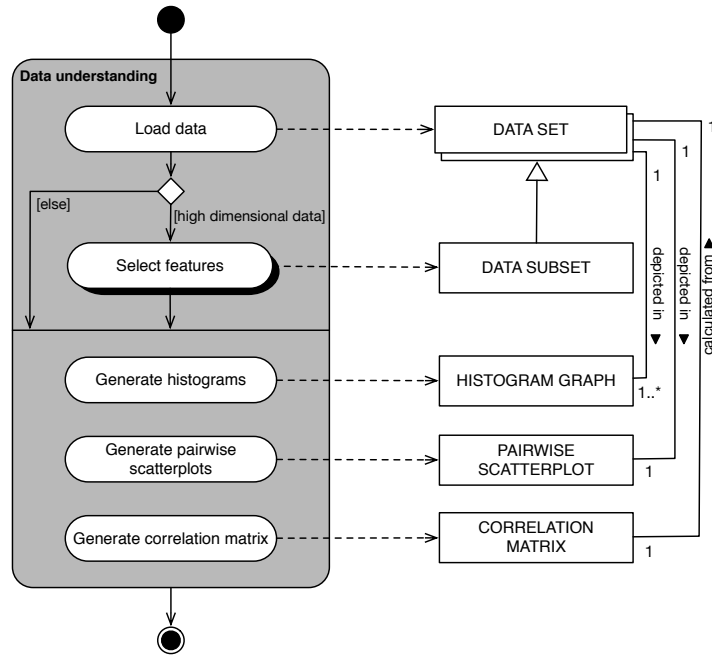


FIGURE 4.1: Process-deliverable-diagram of the data understanding phase

Activity	Sub-activity	Description
Data understanding	Load data	The data scientist loads the raw data into a workbench, this creates the DATA SET that will be explored.
	Select features	If the DATA SET contains a high number of features the data scientist can choose to reduce the dimensionality of the DATA SET. Feature selection is either performed manually using EDA techniques such as CORRELATION MATRICES, PAIRWISE SCATTERPLOTS and / or HISTOGRAM GRAPHS, or selection is performed using a feature selection algorithm.
	Generate histograms	The data scientist generates a HISTOGRAM GRAPH to explore the distribution of the features in the DATA SET.
	Generate pairwise scatterplots	The data scientist generates a PAIRWISE SCATTERPLOT to explore possibilities of separation between features.
	Generate correlation matrix	The data scientist generates a CORRELATION MATRIX to determine which features are informative and check for redundancy between variables.

TABLE 4.1: Activity table of the data understanding phase

Concept	Description
DATA SET	In the context of binary classification problems a DATA SET contain information (features) describing two classes of different entities.
DATA SUBSET	The DATA SUBSET consists of a diminished number of features, usually the most informative variables are selected for the subset.
HISTOGRAM GRAPH	A HISTOGRAM GRAPH is a visual representation of the distribution of the classes over each specific feature.
PAIRWISE SCATTERPLOT	A PAIRWISE SCATTERPLOT visualizes the datapoints for all possible pairs of features on a 2-dimensional canvas. This allows to visually confirm possible areas for class separation.
CORRELATION MATRIX	The CORRELATION MATRIX contains the Pearson correlation values for all possible pairs of features. Features correlated with the class can be considered as informative. Features that are correlated with each other are redundant and must be removed before using the naive bayes classifier.

TABLE 4.2: Concept table of the data understanding phase

4.1.2 Data preparation

The data preparation phase consists of three main activities. The *data set construction* activity entails loading raw data and engineering new features based on this raw data. Feature engineering can be a substantial task but is difficult to capture in a method since it is highly situational. Next the method proceeds with feature selection since not all features tend to have the same amount of informative value. If the data set contains a high number of columns / variables the data scientist can choose to reduce the dimensionality of the data set. Feature selection is either performed manually using EDA techniques, or selection is performed using a feature selection algorithm.

The *feature extraction* activity entails the application of projection methods. Projection methods like principal component analysis are automated feature engineering techniques that aim to best describe the main differentiators of a data set creating a select (low) number of features in the process (dimensionality reduction). Transparency between the outcome variable and the original features may be lost while using a projection technique.

Lastly, the *modeling technique preparation* activity consists of three paths that define preparation steps depending on the model type chosen by the data scientist. When tree and rule models are required due to model transparency concerns no additional preparation steps are necessary since modern algorithm implementations take care of preparation steps internally. Linear models and the probabilistic naive bayes model can be chosen due to performance concerns. Both types require their own conversion steps in order to be able to process the data in the next phase of the DM process. The naive bayes model type e.g. requires redundant features to be removed since they will bias classifier results. Linear model types require input data to be represented in numerical form so transformation steps should be performed as needed e.g.: binarize categorical data. Note however that some concrete algorithm implementations of linear models may perform these steps as part of their internal workings.

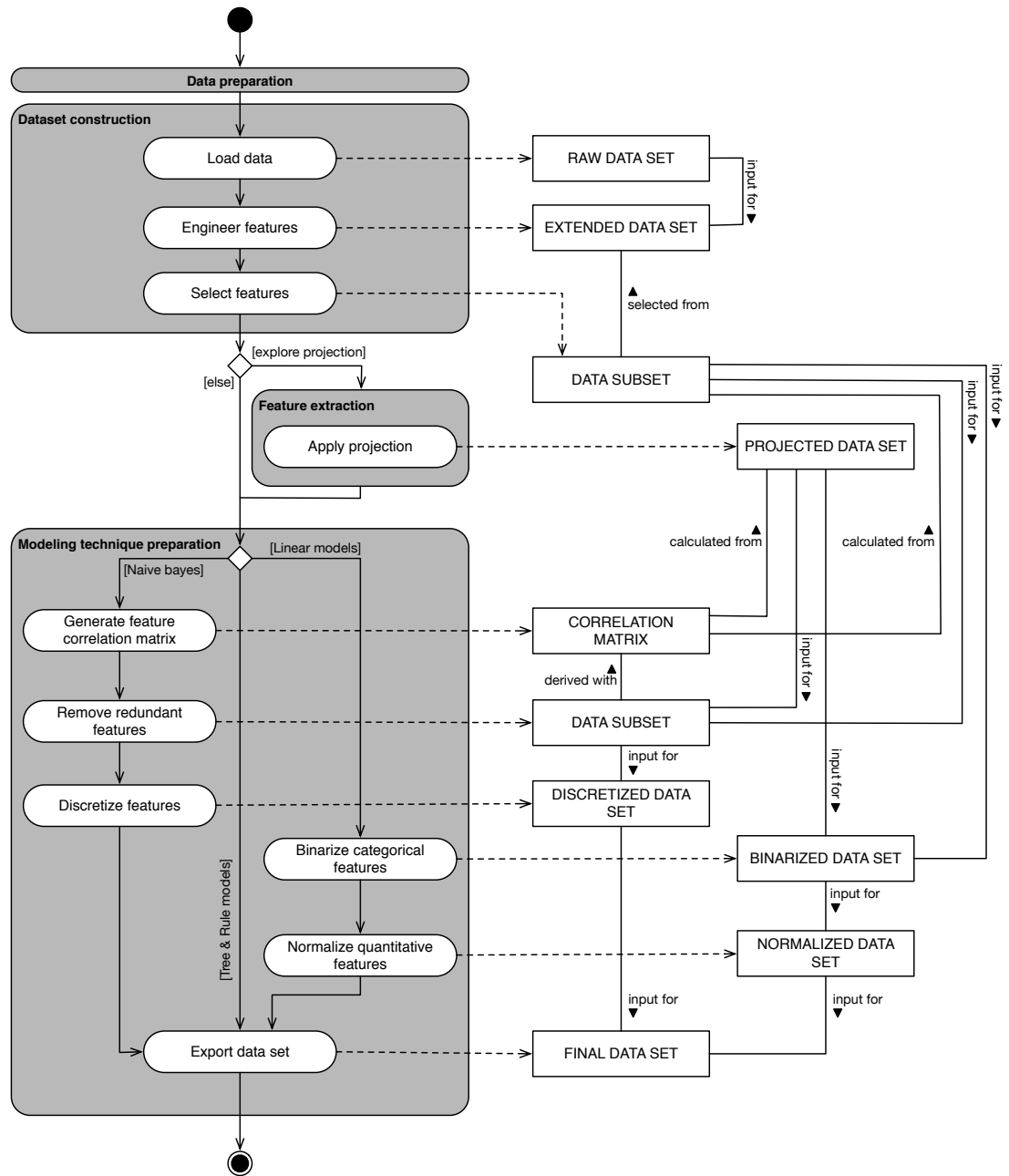


FIGURE 4.2: Process-deliverable-diagram of the data preparation phase

Activity	Sub-activity	Description
Data set construction	Load data	The data scientist loads the raw data into a workbench which delivers a RAW DATA SET.
	Engineer features	The data scientist converts variables with raw data into informative features. For example a column containing dates of birth may be used as input for a new feature that contains age information.
	Select features	The data scientist applies feature selection techniques to the RAW DATA SET.
Feature extraction	Apply projection	The data scientist applies a projection method to the DATA SUBSET.
Modeling technique preparation	Generate feature correlation matrix	The data scientist generates a feature correlation matrix to detect redundant features.
	Remove redundant features	The data scientist removes redundant (correlated) features from the data set.
	Discretize features	The data scientist discretizes the features of the DATA SUBSET resulting in the DISCRETIZED DATA SET.
	Binarize categorical features	The data scientist binarizes the features of the DATA SUBSET resulting in the BINARIZED DATA SET
	Normalize quantitative features	The data scientist normalizes the quantitative features of the DATA SUBSET resulting in the NORMALIZED DATA SET
	Export data set	The data scientist exports the FINAL DATA SET for usage in the modeling and evaluation phase.

TABLE 4.3: Activity table of the data preparation phase

Concept	Description
RAW DATA SET	In the context of binary classification problems a RAW DATA SET contains information (variables) describing two classes of different entities.
EXTENDED DATA SET	The EXTENDED DATA SET represents a RAW DATA SET that is enriched with additional features derived from existing columns or other data sets.
DATA SUBSET	The DATA SUBSET consists of a diminished number of features, usually the most informative features are selected for the subset.
PROJECTED DATA SET	A PROJECTED DATA SET consists of features that are newly generated by a projection method (e.g. principal component analysis).
CORRELATION MATRIX	The CORRELATION MATRIX contains the Pearson correlation values for all possible pairs of features. Features correlated with the class can be considered as informative. Features that are correlated with each other are redundant and must be removed before using the naive bayes classifier.
DISCRETIZED DATA SET	The DISCRETIZED DATA SET contains feature values where original quantitative values are converted to categorical values e.g. the value 5 belonging to the category 0 - 10.
BINARIZED DATA SET	In a BINARIZED DATA SET the various options of a categorical variable are represented as either true or false (binary). This allows for a numeric representation of categorical data which is necessary for linear modeling techniques.
NORMALIZED DATA SET	In a NORMALIZED DATA SET all numeric features are converted to have a similar scale. Differences in scale negatively affect accuracy and performance of popular linear modeling techniques.
FINAL DATA SET	The FINAL DATA SET is used for the modeling and evaluation phase. For binary classification purposes it should contain cleansed data of the most informative features.

TABLE 4.4: Concept table of the data preparation phase

4.1.3 Modeling & Evaluation

The modeling and evaluation method fragment consists of three activities aimed at deriving classification models from data sets. The *search space definition* activity has a route to explore fully automated model (and parameter) selection in analyzing the data set. Currently one experimental implementation exists in the form of Auto-WEKA. Currently, due to the novelty of this technique, the approach should be used to gain initial insight into model types that may perform best on the provided data set.

The application of search strategies like grid and random search is central to the *find optimal parameters* activity. These strategies have a common underlying template that make up the search space. The speed at which an optimal solution is found depends on the strategy chosen to run through the search space. For large spaces combined with large data sets it is recommended to go with random search or bayesian optimization. Furthermore, the structure and accessibility of this approach is in line with the design goal of this research project where we aim to construct a method that enables a user to create optimal models.

Lastly, the *predict and classify* activity defines the steps to apply the model that has been derived to an unseen dataset to gain classification results.

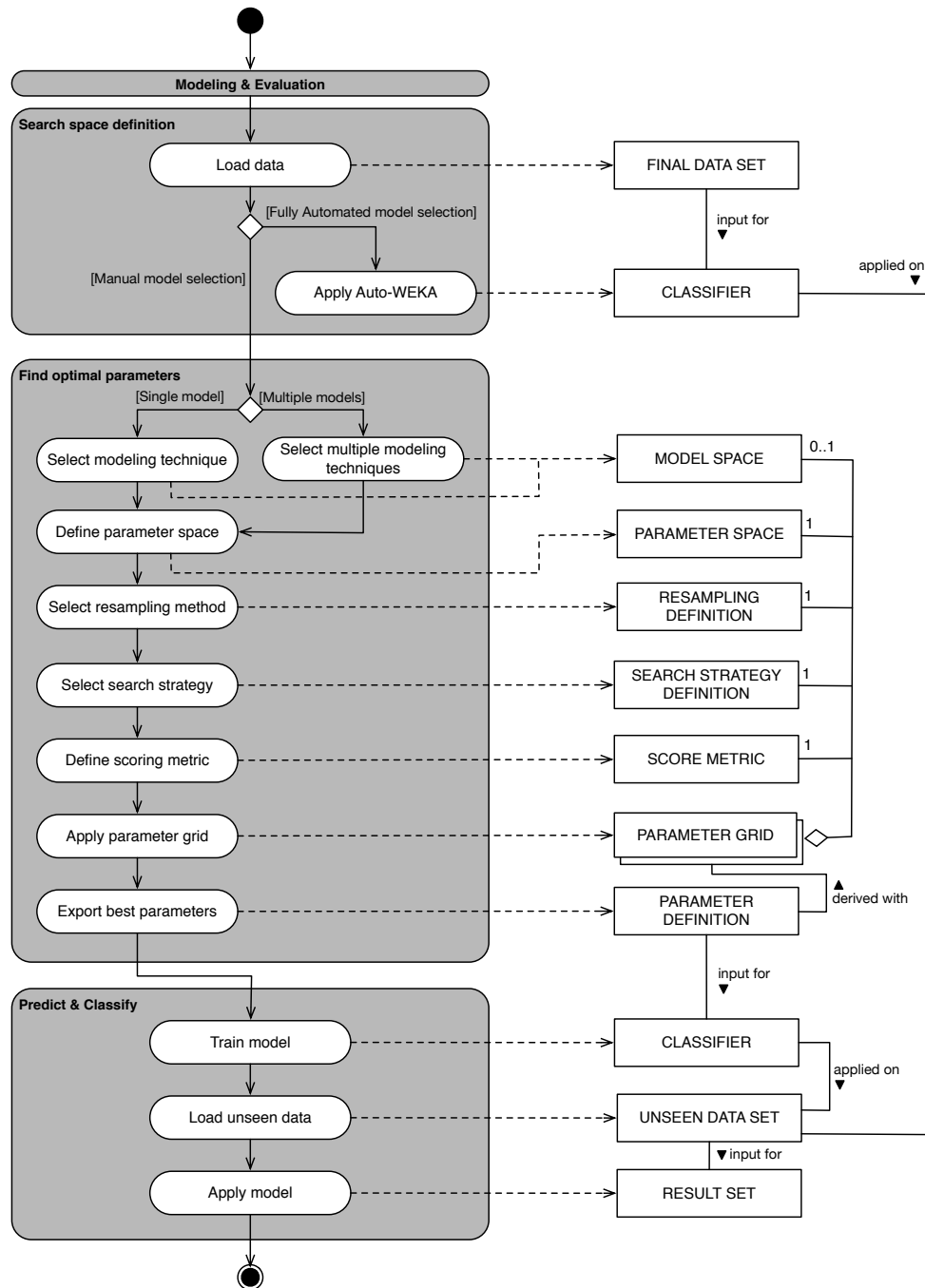


FIGURE 4.3: Process-deliverable-diagram of the modeling & evaluation phase

Activity	Sub-activity	Description
Search space definition	Load data	The data scientist loads the FINAL DATA SET into the workbench.
	Apply Auto-WEKA	The data scientist applies a fully automated model (and parameter) selection tool on the data set.
Find optimal parameters	Select (multiple) modeling technique(s)	The data scientist chooses between a single or multiple model type approach towards training and test runs.
	Define parameter space	The data scientist defines parameter spaces relevant to each modeling type implementation algorithm, this is known as the PARAMETER SPACE.
	Select resampling method	The data scientist decides on a resampling method to support the evaluation process. The hold-out (70-30 split) method or stratified k-fold cross validation can be chosen (usually with 10 folds). For smaller data sets the leave-one-out method is more fitting.
	Select search strategy	The data scientist decides on a search strategy fitting the size and training difficulty of the data set.
	Define scoring metric	The data scientist decides and selects a (combination of) search metric(s) to determine the performance of the selected algorithm(s).
	Apply parameter grid	The data scientist initiates the parameter (and model) search on the FINAL DATA SET.
	Export best parameters	The data scientist exports the best parameters found during the parameter search.
Predict & Classify	Train model	The data scientist trains the model with the best PARAMETER DEFINITION found during the parameter search.
	Load unseen data	The data scientist loads the UNSEEN DATA SET into the workbench.
	Apply model	The data scientist applies the CLASSIFIER to the UNSEEN DATA SET to classify the data.

TABLE 4.5: Activity table of the modeling & evaluation phase

Concept	Description
FINAL DATA SET	The FINAL DATA SET is used for the modeling and evaluation phase. For binary classification purposes it should contain cleansed data of the most informative features.
CLASSIFIER	The CLASSIFIER is a trained model that is used to classify unseen data.
MODEL SPACE	The MODEL SPACE represents the classification algorithms that are used during training and testing runs.
PARAMETER SPACE	The PARAMETER SPACE represents all possible combinations of parameter values that are defined in advance. These settings are specific for algorithm implementations.
RESAMPLING DEFINITION	The RESAMPLING DEFINITION contains the method of choice (hold-out, stratified k-fold cross validation, bootstrapping) for resampling methods and the specific settings (e.g. 70-30 split, 10 folds, leave-one-out).
SEARCH STRATEGY DEFINITION	The SEARCH STRATEGY DEFINITION represents the search strategy of choice: grid search, random search and bayesian optimization. Grid search is exhaustive making it a non-economical choice when exploring various combinations of parameters.
SCORE METRIC	SCORE METRIC's represent the performance of a CLASSIFIER. A combination of the AUC (area under the curve), accuracy, true positive rate (TPR) and false positive rate (FPR) can draw a comprehensive picture of the behavior and performance of a CLASSIFIER.
PARAMETER GRID	The PARAMETER GRID contains pre-defined parameter options for all aspects that are necessary to run a parameter search: MODEL SPACE, PARAMETER SPACE, RESAMPLING DEFINITION, SEARCH STRATEGY DEFINITION and the SCORE METRIC.
PARAMETER DEFINITION	The PARAMETER DEFINITION is the output of the search process and contains the combination parameters that yield the best performance according to the SCORE METRIC's that are chosen.
UNSEEN DATA SET	In this context the UNSEEN DATA SET does not contain any labels and requires classification by a CLASSIFIER.
RESULT SET	After classification the UNSEEN DATA SET is enriched with class data which renders the RESULT SET.

TABLE 4.6: Concept table of the modeling & evaluation phase

4.2 Behapp

In this section we introduce an instance level method that explains how to process location data. In practice the method would be applied as part of the feature engineering activity while preparing the data for analysis. This particular technique was not originally in scope for this research project but was developed in order to process the data for the Behapp project.

4.2.1 Daily average number of places visited

In order to express the social exploration (movement) patterns of subjects in a meaningful way a method was devised to process the location (GPS) data into a concrete measure. We chose to determine the number of unique places visited by each subject on a daily basis. These numbers are summed and then averaged over the total number of days per subject. This results in a measure that counts the average number of places visited by a subject on a daily basis.

In the *load data* activity we extract location data including their timestamps, the latter is needed to separate the datapoints per day. Next in the *calculate distances* activity we slice the data set further down to separate daily repositories of location data for each subject. Each repository is then used to create a distance matrix using the Haversine distance metric (Shumaker and Sinnott, 1984), this specific metric is important since it concerns geographic coordinates.

Lastly, in the *cluster distances* activity the distance matrix is used as input for a density clustering algorithm. These unsupervised algorithms find patterns in datapoints that are in close proximity to each other and assign these points to clusters. The number of clusters found by this algorithm represent the number of notable places that have been visited by a subject. By averaging this over the number of days of data we derive a measure for how many places a subject has visited on average.

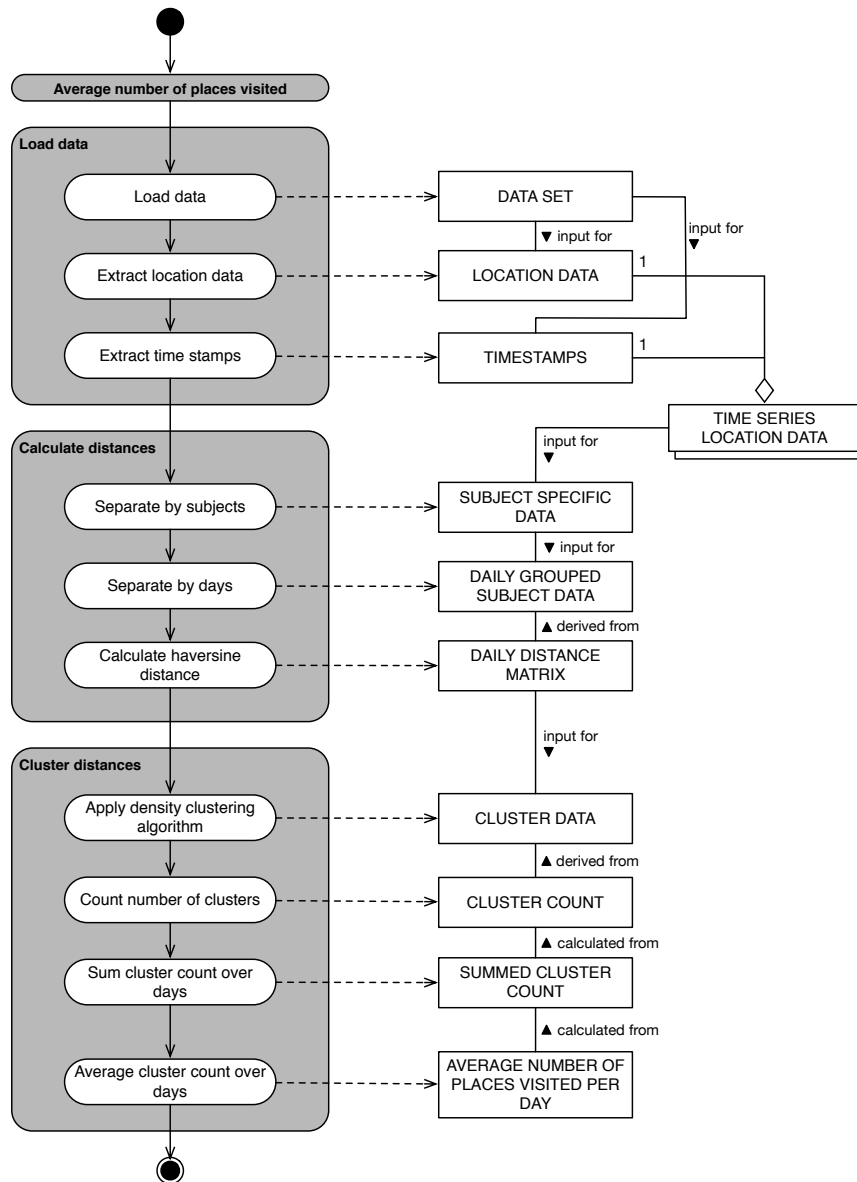


FIGURE 4.4: Process-deliverable-diagram for processing location data

Activity	Sub-activity	Description
Load data	Load data	The data scientist loads the DATA SET into the workbench.
	Extract location data	The data scientist extracts the columns containing the location data e.g. lat/lon coordinates.
	Extract time stamps	The data scientist extracts the column(s) containing timestamps corresponding to the location data.
Calculate distances	Separate by subjects	The data scientist programmatically separates the location data belonging to different subjects.
	Separate by days	The data scientist programmatically separates the location data per day. This yields a DAILY GROUPED SUBJECT DATA set.
	Calculate Haversine distance	The data scientist calculates pairwise distances between daily location data points using the Haversine distance metric.
Cluster distances	Apply density clustering algorithm	The data scientist applies a density clustering algorithm (e.g. DBSCAN or OPTICS) to the DAILY DISTANCE MATRIX. This results in an overview of CLUSTER DATA.
	Count number of clusters	The data scientist determines the CLUSTER COUNT based on the CLUSTER DATA by counting the number of clusters that are returned.
	Sum cluster count over days	The data scientist programmatically sums the number of clusters for each day of data belonging to each specific subject.
	Average cluster count over days	The data scientist averages the number of clusters over all days of data belonging to a specific subject. This results in the AVERAGE NUMBER OF PLACES VISITED PER DAY.

TABLE 4.7: Activity table for processing location data

Concept	Description
DATA SET	In the context of binary classification problems a DATA SET contains features describing two classes of different entities.
LOCATION DATA	LOCATION DATA refers to geographic coordinate systems used to express geospatial data, a common format is the latitude / longitude coordinate system.
TIME STAMPS	TIME STAMPS represent the time or datetime of the creation of a location record.
TIME SERIES LOCATION DATA	The LOCATION DATA and corresponding TIME STAMPS combined make up the TIME SERIES LOCATION DATA.
SUBJECT SPECIFIC DATA	A subject specific sliced down repository of TIME SERIES LOCATION DATA.
DAILY GROUPED SUBJECT DATA	A daily sliced down repository of SUBJECT SPECIFIC DATA.
DAILY DISTANCE MATRIX	The DAILY DISTANCE MATRIX contains pairwise Haversine distances between all coordinates of a given day per subject.
CLUSTER DATA	The CLUSTER DATA contains information about the cluster assignment of each coordinate. This result is achieved by applying a density clustering algorithm like DBSCAN or OPTICS to the DAILY DISTANCE MATRIX.
CLUSTER COUNT	The total number of clusters that are found for a given day.
SUMMED CLUSTER COUNT	The summed number of clusters over all days for a given subject.
AVERAGE NUMBER OF PLACES VISITED PER DAY	The AVERAGE NUMBER OF PLACES VISITED PER DAY is a metric to express the social exploration efforts of subjects.

TABLE 4.8: Concept table for processing location data

Chapter 5

Method evaluation

As stated in Chapter 1 we aimed to design a method which helps to create optimal models. In this chapter we will evaluate the effectiveness of the method fragments as presented in Chapter 4 by embedding them in a formal experiment.

We will apply the experiment process as outlined by Wohlin et al. (2012). The process entails a scoping, planning, operation, analysis & interpretation and lastly, a presentation phase. Each phase has its own deliverable: goal definition, experiment design, experiment data, conclusions and experiment report. We address these deliverables and the resulting experiment in sections 5.1 to 5.4 where the last three deliverables are combined in one section.

5.1 Goal definition

According to Wohlin et al. (2012) the scope of an experiment is derived from determining the goals of the experiment. We use the goal template by Basili and Rombach (1988) to formulate the goal definition of this experiment:

Analyze **machine learning methods**

For the purpose of **evaluating the aforementioned methods**

With respect to their **performance in terms of accuracy and the area under the curve**

From the point of view of the **researcher / user**

In the context of **supervised binary classification tasks**

We will compare the quality of models in two conditions. In condition number one no data preparation steps will be applied to our data and the algorithms belonging to various model types will be run at their default parameter settings. In the second condition our data preparation and modeling & evaluation method fragments will be applied to the data, see Appendix A for our configuration of the method fragments as outlined in Chapter 4. Due to the fully integrated nature of these method fragments we will henceforth refer to our method as the *integrated ML method*. The first condition will be referred to as the *plain ML method*.

To ensure similar testing conditions the single model route will be used from the modeling & evaluation method fragment. Additionally the experiment will be performed as a multi-object study where both the Iris and the Behapp data sets will be used for testing.

5.2 Experiment design

The design of an experiment is the outcome of the planning phase in the formal experiment process. We will address the following concepts to ensure reliable results: design type, hypothesis formulation, variables selection, object selection, instrumentation and validity evaluation.

Design type

The design of this experiment is of the one factor type with two treatments. The factor is the performance of the models that are generated using the plain and integrated modeling methods (treatments).

Hypothesis

We hypothesize that the application of the integrated ML method will result in models that deliver improved performance over models created using the plain ML method:

Null: Models generated using the plain ML method and the integrated modeling method will perform similarly

Alternative: Models generated using the integrated ML method will have improved performance over plain models.

Variables & Object selection

In this experiment the aforementioned modeling methods (plain / integrated) will be our independent variable. We will determine the effect of the methods on the performance of the models that are generated. For each modeling method we will select one algorithm belonging to the following model types: tree, linear and ensemble. The performance will be measured by a combination of the following metrics: accuracy and the area under the curve. The objects in this experiment are represented by two data sets: Iris and Behapp. These data sets will be used to generate and test the models that will be used to evaluate the modeling methods. See Figure 5.1 for a graphical overview of the experiment model.

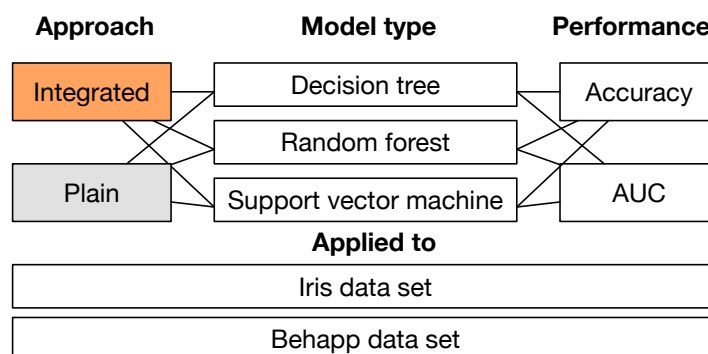


FIGURE 5.1: Graphical overview of the experiment model

Instrumentation

Since the experiment will be performed without subject interaction the only instrumentation items needed are the measurement instruments themselves. We will use the python scientific stack (scipy, pandas and scikit-learn) to

prepare, run and measure the outcomes of the experiment.

Validity evaluation

Conclusion validity is concerned with ensuring that the outcome and the treatment have a valid statistical relationship. Since optimization strategies in general show small increments in performance (Thornton et al., 2013) is yet unknown how strong of a statistical relationship can be established between our method and the outcomes.

Internal validity is concerned with ensuring that any observed effect is in direct relation with the treatment that is applied. We expect that internal validity is ensured since we do not have any interaction with human subjects and only work with stateless objects. However according to the *no free lunch theorem* (NFLT) the representation of data in data sets is related to the mathematical constructs of various algorithms and modeling types. The theorem explains that for this reason the existence of one superalgorithm is not possible (Ho and Pepyne, 2002). We expect the impact on this experiment to be minimal since we are using algorithms belonging to different model types in our experiment setup.

Construct validity is concerned with the connection between theory and the observed effect. This validity type is ensured by having constructed the method from well tested theories and practical ML knowledge bits, see Chapter 3.

Last, external validity is concerned with the generalizability of the outcomes found in the experiment. Due to time constraints the methods are tested on a small set of objects which poses a threat to the generalizability of the outcomes to larger and more complex data sets.

5.3 Data description

5.3.1 Iris data set

The Iris data set, as introduced in Chapter 3 is originally a three class data set that contains four features describing petal measurements of the three classes of flowers (Setosa, Versicolor and Virginica). For each class there are 50 observations (instances). For this experiment we converted this data set to a two class data set to by removing the datapoints belonging to the Setosa class. We chose for this class because it is already linearly separable from the other two classes, thus it would be easy to separate from the other two classes regardless of the methods applied.

5.3.2 Behapp data set

The Behapp data set is a two class data set about a group of 17 participants, 4 patients and 13 controls. The set contains 17 records, one for each participant, with 21 features describing various explorative and social acts. The data shows that social media (e.g. Twitter and Facebook) and text messaging services (e.g. MMS and SMS) are mostly left unused by our participants, thus we expect these to be filtered out during feature selection. On the other hand, datapoints related to location services, bluetooth detection and call logs are consistently measured and expected to carry most of the informative value.

5.4 Data & Interpretation

See Table 5.1 for an overview of the experiment data. The table holds the accuracy and AUC scores for the Iris and Behapp data sets. For both data sets three different model types have been applied using the plain and integrated modeling methods.

		Plain		Integrated	
		Accuracy	AUC	Accuracy	AUC
Iris	Decision tree	0.939	0.939	0.940	0.948
	Random forest	0.950	0.964	0.950	0.985
	Support vector machine	0.959	0.994	0.960	0.994
Behapp	Decision tree	0.662	0.437	0.765	0.806
	Random forest	0.762	0.552	0.882	0.806
	Support vector machine	0.762	0.5	0.824	0.706

TABLE 5.1: Classification score overview of the Iris and Behapp data set

We observe an improvement in the accuracy and AUC scores for the majority of the cases while using the integrated approach. However, the improvements gained on the Iris data set are only minimal. This can be explained by the fact that the *versicolor* and *virginia* classes of datapoints are almost linearly separated which does not leave a lot of room for improvement. Nonetheless for both the decision tree and the random forest we observe an improvement of 0.8 to 2 percent points which falls in line with the goal of the integrated approach of delivering optimized models.

Next concerning the Behapp data set we observe more dramatic improvements in accuracy and AUC scores while using the integrated ML method. Compared to the Iris data set the Behapp data set contains more uninformative features (22 versus 4) which negatively influence the learning process of the models that are applied. We observe a direct increase in classification performance after feature selection limiting the data subset to the two most informative features. Further improvements are gained using the grid search parameter optimization strategy which iterates over 8 to 144 candidate parameter settings.

To conclude we observe that the integrated ML method realizes improved classifier performance finding both room to improve well working classifiers as well as uncovering informative patterns in data sets to achieve optimal performance. See Appendix C and Appendix D for a complete overview of the notebooks containing the experiment source code and output.

Chapter 6

Behapp data insights

As related to our overarching research goals, in this chapter we present the insights gained into the Behapp data set during our data preparation and experimentation efforts. We will start by documenting our data wrangling and data preparation activities in section 6.1. Next, in section 6.2 we determine what the most informative features are. Furthermore, in section 6.3 we generate a tree model visualizing the decision boundaries of the final model. Lastly, in section 6.4 we will convert the features into social profiles according to the social profiling technique by Eskes et al. (2016). The technique serves as a projection to express the social communication and exploration acts of subjects.

6.1 Data wrangling

Data wrangling or also known as data munging is the terminology used for cleaning, transforming and unifying raw data for easy consumption further down the DM process. As demonstrated in Appendix E the actions performed are specific to the context of the data sets in scope. Because of this we decided not to attempt to model these actions even though they can be considered as data preparation steps.

Behapp data is originally provided as time series data indicating when a specific event has taken place. These events can be e.g. an incoming call, location update or the detection of a bluetooth device. The events are stored in a relational database where each event is related to a specific participant. The number of events can run into the hundreds or thousands of rows for participants that have been monitored for at least more than one month.

We start by merging this data with the participant database that indicates whether a participant belongs to the control or patient group. Next we perform some basic transformations to get the events overview in a state where we can start summarizing the data. We rename columns for better clarity and select only the columns that are relevant to our analysis.

With the events overview in a finalized state we initialize and automated count for each event type that is known to the system. For most event types this is enough to define the event type as a measure. For example, the number of incoming and outgoing calls can be directly used by simply counting the number of instances. This count is performed on a participant by participant basis and is referred to as our general overview.

The overview is then supplemented with features that need additional processing instead of a simple count. One example is demonstrated in Chapter 4 where we present our feature engineering method to derive the average number of places visited per day (per participant) based on gps

location data. Another example is the determination of the size of a participants' social circle, this is calculated by extracting the number of unique phone numbers from the event data belonging to a specific participant.

The last important step in our data wrangling efforts is the standardization of data over time. There are substantial differences between the participants concerning the amount of time they have been monitored by Behapp. In order to make a fair comparison the measurements are standardized to a common length of time. Since the range can vary from ten days to one year we choose to standardize all data over one day. First we establish the runtime of the app per participant in two ways: 1) by calculating the number of days between the first and last event received and 2) by calculating the unique number of days in the database for a given participant. The second calculation is more accurate when there are gaps in the data, meaning that there are periods where no data is recorded / received at all.

Finally, all measurements are divided by the number of active days which yields a standardized overview of measurements where participants can be compared to each other. See Appendix B for an overview of this data set. Note that we have only discussed the highlights of our data wrangling efforts, for more details please see Appendix E.

6.2 Data preparation - Feature selection

Based on our requirement for model transparency we choose to model the data using tree models. The data preparation method fragment as defined in subsection 4.1.2 shows that this path only requires the application of feature selection as a preparation step.

We applied this technique to the Behapp data set as part of the experiment conducted in Chapter 5. Appendix D shows that the following two features are found to be most informative in separating patients from controls: *bluetooth device diversity* and *unique contacts*. Bluetooth device diversity refers to the unique number of bluetooth devices that a subject has encountered during the monitoring study. Unique contacts is similar but measures the unique contacts encountered in call and sms usage.

6.3 Modeling - Tree model

Finally, we visualized the tree model generated during the experiment, it is depicted in Figure 6.1. The model was generated using a grid search iterating over 8 candidate parameter settings with the best combination of settings reaching a classification accuracy of 0,824% (Appendix D).

The model shows the bounds of the bluetooth and contacts features and how these are related to the classification of the two subject groups. We see that the majority (3) of our patients are currently classified as such when their number of unique contacts ranges from 21 to 27.5 contacts. One of our patients is a (known) extreme outlier in the sense of the exploration and communication efforts which explains the additional nodes separating subjects with a bluetooth device diversity lower than 976 and a higher upper limit of the number of unique contacts (435.5). We should stress that the data presented here is very preliminary. We expect that the relationships established will shift over time as we collect more data.

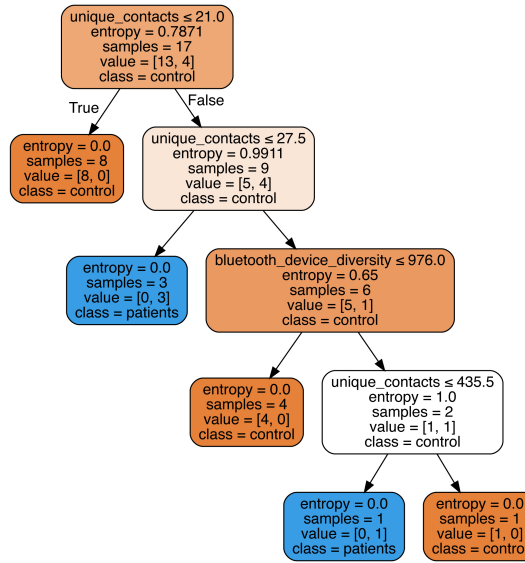


FIGURE 6.1: Decision tree model of the Behapp data set

6.4 Social Profiling

Our social profiling approach is based on the work by Eskes et al. (2016) who has been involved with the Behapp project during an earlier phase of the initiative. The author validated the use of metrics gained through smartphones in relation to dimensions like *social exploration* and *communication*.

Next the author devised a scoring model to summarize a selection of the 23 metrics of the Behapp application on the two dimensions as mentioned above. This enables us to express the *sociability* of a subject on a 2-dimensional plane. The scoring model works as follows:

- First, all metrics are standardized using z-score standardization over the whole population
- Second, the resulting z-scores are converted to a more comprehensible number space which yields numbers between 0 and 100
- Third, metrics related to communication efforts are grouped and averaged resulting in the communication score
- Fourth, metrics related to social exploration efforts are grouped and averaged resulting in the social exploration score

Figures 6.2 and 6.3 depict the social profiles of our participants. The first plot is implemented according to the original model as devised by Paul Eskes. The second plot has been extended with additional features that were found to be relevant to both dimensions. The original Eskes plot is composed of the following features:

- Communication score
 - Incoming calls
 - Outgoing calls

- Call duration
- Social exploration score
 - Places visited daily
 - Bluetooth devices detected

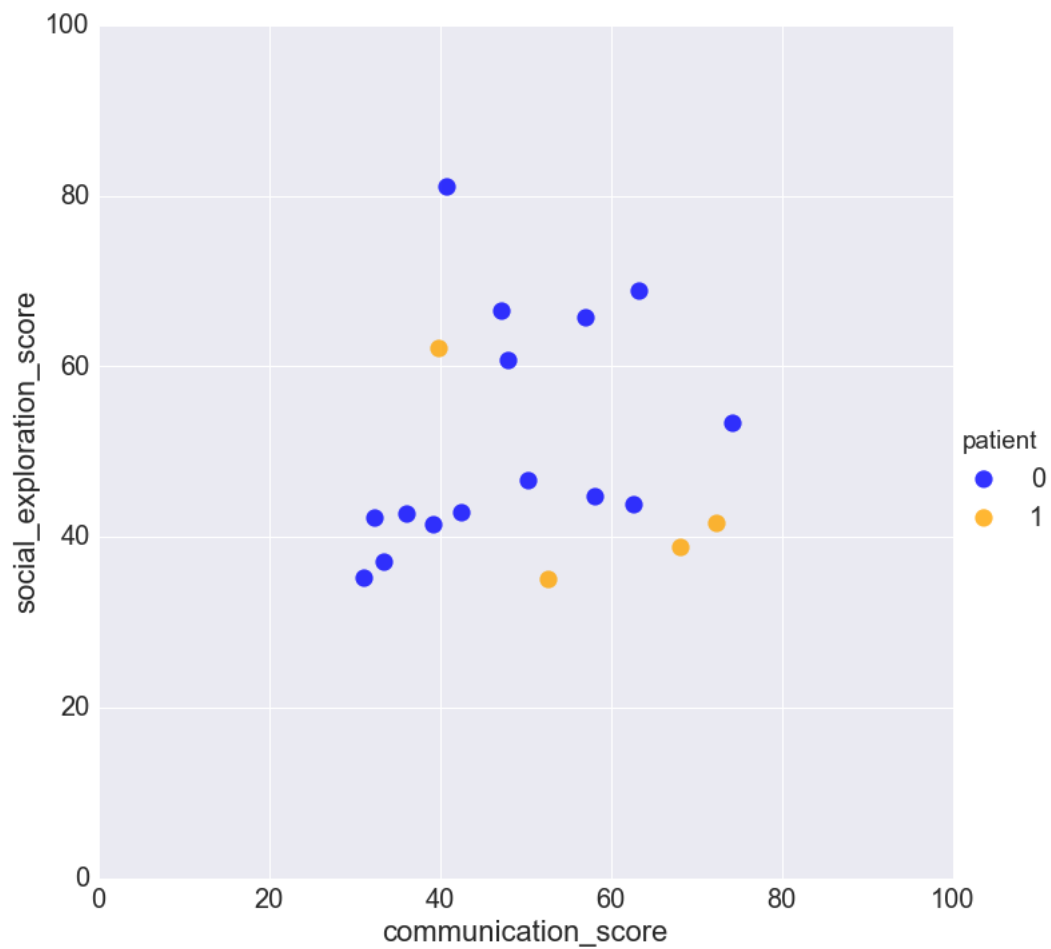


FIGURE 6.2: Sociability score plot of Behapp participants - Original Eskes

The modified Eskes plot has been extended with the newly added features marked in bold:

- Communication score
 - Incoming calls
 - Outgoing calls
 - **Missed calls**
 - Call duration
 - **Unique contacts**
 - **Whatsapp activity**
- Social exploration score

- Places visited daily
- Bluetooth devices detected
- **Bluetooth device diversity**

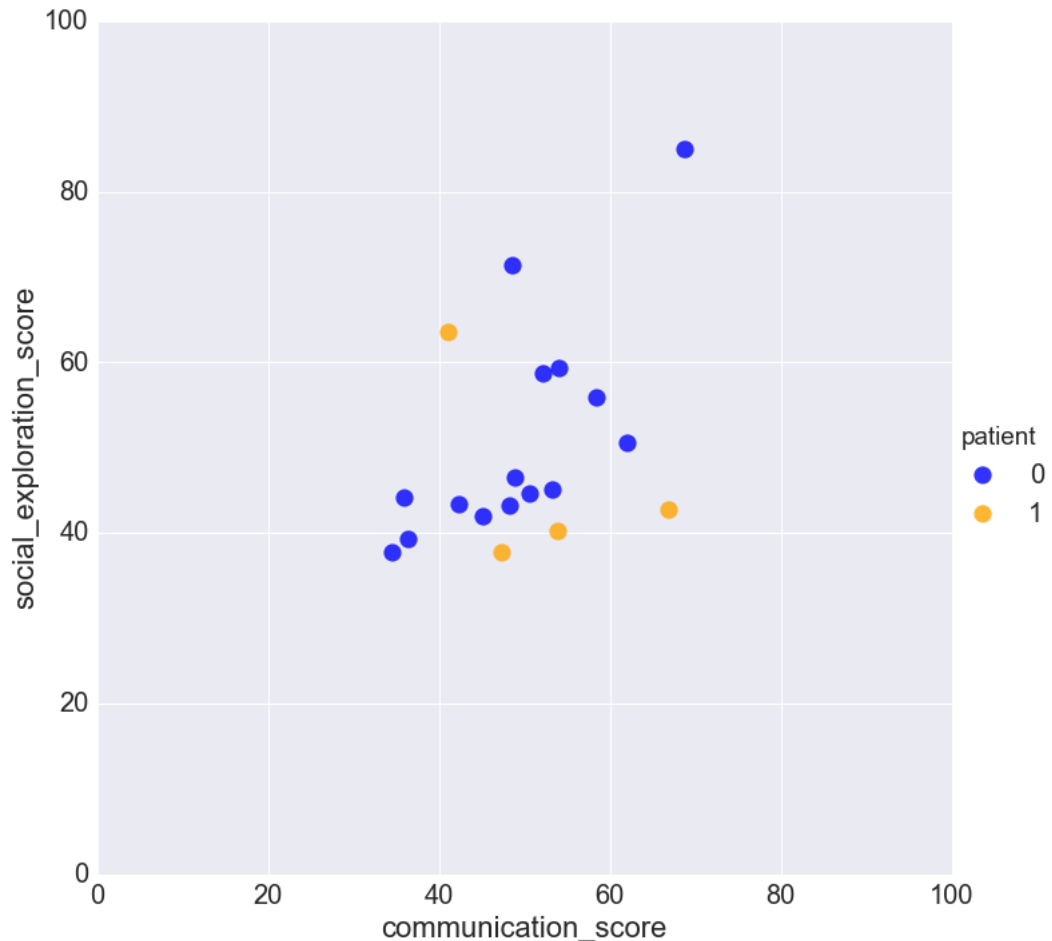


FIGURE 6.3: Sociability score plot of Behapp participants - Modified Eskes

Both plots highlight an interesting pattern, we observe that the patients are either highly communicative and limited in their social exploration efforts or the other way around. During an informal evaluation session it was noted by one of the project supervisors that the mark of a patient is perhaps to be found in the skewed ratio between the communication and exploration efforts of a subject. Again we need to stress that the data presented is preliminary and any patterns found may change over time.

Chapter 7

Conclusion

In this chapter we will present our conclusion to the main research question:

How can a domain independent method be developed to guide the process of constructing transparent machine learning models?

We formulated a number of sub-questions to structure the research process, the conclusions to these questions are discussed first after which we conclude with an answer to the main research question.

7.1 Conclusion of sub-questions

Questions one to three are based on theoretical foundations derived from the body of literature on data mining and machine learning. Question four is based on theory on method engineering and also covers the realization of the main deliverable of this thesis: the integrated ML method (see Appendix A).

Question one

Which data mining process models are available and how do they fit machine learning techniques?

We briefly reviewed six data mining process models and explored their properties concerning prevalence in business and academics, level of depth, and technical detail. We found that the CRISP-DM process model exhibited the strongest properties concerning the aforementioned factors. However the human centered approach was found to be a strong second contender lacking in prevalence in business and academics.

In the CRISP-DM context, the integrated classification method fragments as presented in chapter 4 represent specialized tasks. Although not visualized these tasks can be mapped to the generic tasks in the CRISP-DM hierarchy adding to the comprehensible value of the method.

Question two

What are the main concepts involved in constructing a supervised machine learning system for classification tasks?

We explored machine learning concepts over a broad spectrum touching upon data understanding, preparation and modeling techniques. First, we

found that presenting every algorithm implementation would be counter-productive in communicating a general strategy. Instead we opted to discuss modeling techniques on the level of model types (e.g. tree, linear etc.) outlining the frequently cited algorithms within each type. Next, for each model type we derived common properties in terms of requirements to data (pre) processing. These are addressed in section 3.3.1. Last we discuss various techniques to evaluate the informative value of data sets and explain how they impact classification performance.

a) Which of these concepts have aspects that support (automated) configuration and optimization and how does this work?

We distinguish between two automated optimization approaches. First we explain the concept of feature selection as both a manual and an algorithm assisted technique. Next we researched (hyper) parameter optimization strategies and found techniques like bayesian optimization and random search that enable effective and economic searches for parameters that deliver optimal models. The concrete implementation of these techniques may vary, some machine learning toolkits provide abstract interfaces to easily define and run a search task. Outside of these toolkits the search process will have to be specified and thus programmed by hand.

Question three

What are transparent models and which techniques are suitable for this type of modeling?

In section 1.2 we identify the need for transparent models when used in a medical context. Transparent models allow the user to directly relate input data to the decision (output) of a model. Both tree and rule models are expressive by nature and fall in the category of transparent model types.

Most other techniques however do not offer any kind of insight. The research shows that these black box techniques are often found to be superior in performance compared to transparent techniques. We found methods to extract rules from these black box models but they are criticized for oversimplifying the model losing information in the process.

Last, we found a new method aimed to even the playing field between transparent and black box models concerning classification performance while at the same time offering a detailed overview of input / output relationships (Caruana et al., 2015). Unfortunately due to incompatibility issues the software behind this technique could not be further explored in this thesis.

Question four

How can the concepts that are found be modelled into a method?

We chose to model our method using the method engineering approach as refined by Weerd and Brinkkemper (2008). The approach outlines the creation and union of activity and class diagrams that express flows of activities and the deliverables that are involved in these activities.

See Appendix A for the resulting method of this research project. The method is structured as three separate method fragments that each correspond to one of the following CRISP-DM main phases: data understanding,

data preparation and modeling. More concretely the method fragments address 1) the relationships between model types and (pre) processing steps and 2) the application of search strategies as an integral part of the modeling exercise, hence the term integrated ML method.

7.2 Conclusion of main question

How can a domain independent method be developed to guide the process of constructing transparent machine learning models?

We set out with the overarching objective of this thesis to analyze a medical data set containing measurements gathered by use of mobile devices. The goal was to research and develop predictive and explanatory models for the Behapp initiative. In order to realize this goal a method had to be devised to undertake this classification problem. First we needed to find the right base structure to provide a sense of familiarity and boost comprehension of the proposed method, this structure was found in the CRISP-DM process model. Next we identified various common threads in the practice of supervised binary classification tasks, we expressed these threads using the method engineering approach by Weerd and Brinkkemper (2008). We found automated feature selection approaches and parameter search strategies to be of notable importance to classifier performance and designated these as central activity components. Next, we evaluated the method and found that adhering to the base steps consistently increases the performance roof of classifiers generated using our method. In sum, we can conclude that the insights gained from answering the sub-questions enabled us to achieve the goal of developing a domain independent method to guide the process of constructing transparent machine learning models. Last, we applied transparent modeling techniques to the Behapp data set and uncovered patterns separating patients from controls. Furthermore we modernized and re-applied the social profiling model of Eskes et al. (2016) and found additional patterns concerning diffences in behavior between patients and control participants.

Chapter 8

Discussion

This chapter is divided in two sections, in section 8.1 we will address the limitations of our research. In section 8.2 possibilities for future research are discussed.

8.1 Limitations

We distinguish between three categories concerning the limitations of this research project: instrumental limitations, research design limitations and Behapp data limitations. We will elaborate on these limitations below.

First, our instrumental limitations are software issues related to the machine learning toolkits used to generate and test our models. In section 3.5 we discussed the work of Lou et al. (2013) where the authors explain to have developed a new modeling technique for deriving accurate and transparent models. We intended to apply the technique to our own data sets however we failed to get the software to work. This limited our options for using transparent modeling techniques during the experiment and the Behapp data exploration phase. Next we found discrepancies between models based on similar algorithms but using different machine learning toolkits. Furthermore the toolkits were each found to be strong in their own respective areas usually omitting model types and / or functionalities like resampling and parameter search strategies. This limited our options for the design of the experiment, e.g. rule models could not be tested since they are not included in scikit-learn. On the other hand scikit-learn offered excellent interfaces for parameter searching which in turn were limited in functionality in the *Waikato environment for knowledge analysis* (WEKA) (Hall et al., 2009).

Second, as explained in our experiment validity analysis (section 5.2) the time allotted did not allow for extensive testing over a large number of different datasets. Instead we decided to proceed with the Behapp dataset and the frequently cited Iris data set for our experiment. This means that generalization of the results to larger and more complex datasets is currently unconfirmed. For future analysis it would be interesting to see how the method holds up while using larger (more rows) and high dimensional (more columns) data sets.

Lastly, the contents of the Behapp data set brought along their own set of challenges. First at the start of this research project we discovered the loss of an important metric. Due to improvements in security measures the Behapp mobile application lost the ability to monitor Whatsapp, a popular messaging service in the Netherlands. Next the data set contained a small

number of participants, 13 controls and 4 patients. In multiple cases participants were only registered for the minimum number of days (10 to 14) in order to derive meaningful patterns from their data. This limits the ability, for now, to derive conclusive insights from the data.

8.2 Future research

The research conducted has triggered possibilities for future research both from the perspective of the integrated method and the Behapp initiative.

First, the problem space of our research could be broadened to cover cases outside of the domain of supervised binary classification, e.g. multiclass, regression and image analysis problems. Method fragments could be created to deal with (sub)cases in the aforementioned domains. Next the structures defined in these methods could be used for the development / enhancement of data mining tools. Auto-WEKA is an example of such a tool but, but follows a very rigid method (Thornton et al., 2013) in their goal to make machine learning accessible to a broader public. For example, the tool follows a pre set path of actions and tasks and does not support embedding domain knowledge during the DM process. From our own experience we identify a need for sophisticated tools that offer simplified access to advanced ML techniques while retaining the ability to embed domain knowledge in the data mining exercise.

Furthermore, the initial data from the Behapp initiative showed promising underlying patterns in separating patients from control groups. The application is entering a stage where it should be deployed on a larger scale to derive statistically robust social profiles for the participant classes that are part of the studies. However it yet unknown whether it will be feasible to detect patterns that indicate social withdrawal in participants. Furthermore next to the existing features research should be performed into the feasibility of additional measurements like e.g. mood determination based on voice analysis. Lastly, we expect the mobile application to undergo various evolutions in the nearby future. For the sociability model (Eskes et al., 2016) to stay relevant it is important to continuously evaluate what is expressed and how the features measured correspond to the communication and exploration dimensions that are part of the model.

Bibliography

- Agrawal, Rakesh and John C. Shafer (1996). "Parallel mining of association rules". In: *IEEE Transactions on Knowledge & Data Engineering* 6, pp. 962–969.
- Allahyari, Hiva and Niklas Lavesson (2011). "User-oriented Assessment of Classification Model Understandability." In: *SCAI*, pp. 11–19.
- Amoebe, A. (2015). *Making sense of principal component analysis, eigenvectors & eigenvalues*.
- Basili, Victor R. and H. Dieter Rombach (1988). "The TAME project: Towards improvement-oriented software environments". In: *Software Engineering, IEEE Transactions on* 14.6, pp. 758–773.
- Bellazzi, Riccardo and Blaz Zupan (2008). "Predictive data mining in clinical medicine: current issues and guidelines". In: *International journal of medical informatics* 77.2, pp. 81–97.
- Bergstra, James and Yoshua Bengio (2012). "Random search for hyper-parameter optimization". In: *The Journal of Machine Learning Research* 13.1, pp. 281–305.
- Blattberg, Robert C. "Byung-Do Kim, and Scott A. Neslin (2008), Database Marketing: Analyzing and Managing Customers". In: *International Series in Quantitative Marketing*.
- Bolstad, William M. (2013). *Introduction to Bayesian statistics*. John Wiley & Sons.
- Brachman, Ronald J. and Tej Anand (1996). "The process of knowledge discovery in databases". In: *Advances in knowledge discovery and data mining*. American Association for Artificial Intelligence, pp. 37–57.
- Breiman, Leo (1996). "Bagging predictors". In: *Machine Learning* 24.2, pp. 123–140.
- Brinkkemper, Sjaak (1996). "Method engineering: engineering of information systems development methods and tools". In: *Information and software technology* 38.4, pp. 275–280.
- Brinkkemper, Sjaak, Motoshi Saeki, and Frank Harmsen (1999). "Meta-modelling based assembly techniques for situational method engineering". In: *Information Systems* 24.3, pp. 209–228.
- Brownlee, Jason (2014a). *Discover Feature Engineering, How to Engineer Features and How to Get Good at It*.
- (2014b). *Understand your problem and get better results using exploratory data analysis*.
- Caruana, Rich et al. (2015). "Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 1721–1730.
- Chapman, Pete et al. (1999). "The CRISP-DM process model". In: *The CRIP-DM Consortium* 310.
- Chen, Yen-Liang et al. (2005). "Market basket analysis in a multiple store environment". In: *Decision Support Systems* 40.2, pp. 339–354.

- Cortes, Corinna and Vladimir Vapnik (1995). "Support-Vector Networks". English. In: *Machine Learning* 20.3, pp. 273–297. ISSN: 0885-6125. DOI: 10.1023/A:1022627411411. URL: <http://dx.doi.org/10.1023/A%3A1022627411411>.
- Cortez, Paulo and Mark J. Embrechts (2013). "Using sensitivity analysis and visualization techniques to open black box data mining models". In: *Information Sciences* 225, pp. 1–17.
- Delen, Dursun, Glenn Walker, and Amit Kadam (2005). "Predicting breast cancer survivability: a comparison of three data mining methods". In: *Artificial Intelligence in Medicine* 34.2, pp. 113–127.
- Domingos, Pedro (2012). "A few useful things to know about machine learning". In: *Communications of the ACM* 55.10, pp. 78–87.
- Domingos, Pedro and Michael Pazzani (1997). "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss". English. In: *Machine Learning* 29.2-3, pp. 103–130. ISSN: 0885-6125. DOI: 10.1023/A:1007413511361. URL: <http://dx.doi.org/10.1023/A%3A1007413511361>.
- Dougherty, James, Ron Kohavi, and Mehran Sahami (1995). "Supervised and unsupervised discretization of continuous features". In: *Machine learning: proceedings of the twelfth international conference*. Vol. 12, pp. 194–202.
- Eskes, Paul et al. (2016). "The sociability score: App-based social profiling from a healthcare perspective". In: *Computers in Human Behavior* 59, pp. 39–48.
- Fayyad, Usama, Gregory Piatetsky-Shapiro, and Padhraic Smyth (1996). "From data mining to knowledge discovery in databases". In: *AI magazine* 17.3, p. 37.
- Fisher, Ronald A. (1936). "The use of multiple measurements in taxonomic problems". In: *Annals of eugenics* 7.2, pp. 179–188.
- Flach, Peter (2012). *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press.
- Freitas, Alberto, Altamiro Costa-Pereira, and Pavel Brazdil (2007). "Cost-sensitive decision trees applied to medical data". In: *Data Warehousing and Knowledge Discovery*. Springer, pp. 303–312.
- Freund, Yoav and Robert E. Schapire (1995). "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Computational learning theory*. Springer, pp. 23–37.
- Gertosio, Christine and Alan Dussauchoy (2004). "Knowledge discovery from industrial databases". In: *Journal of Intelligent Manufacturing* 15.1, pp. 29–37.
- Giudici, Paolo (2005). *Applied data mining: statistical methods for business and industry*. John Wiley & Sons.
- Guyon, Isabelle and André Elisseeff (2003). "An introduction to variable and feature selection". In: *The Journal of Machine Learning Research* 3, pp. 1157–1182.
- Hall, Mark et al. (2009). "The WEKA data mining software: an update". In: *ACM SIGKDD explorations newsletter* 11.1, pp. 10–18.
- Hallinan, Jennifer (2014). *Assessing and Comparing Classifier Performance with ROC Curves*.
- Hevner, R. Alan von et al. (2004). "Design science in information systems research". In: *MIS quarterly* 28.1, pp. 75–105.

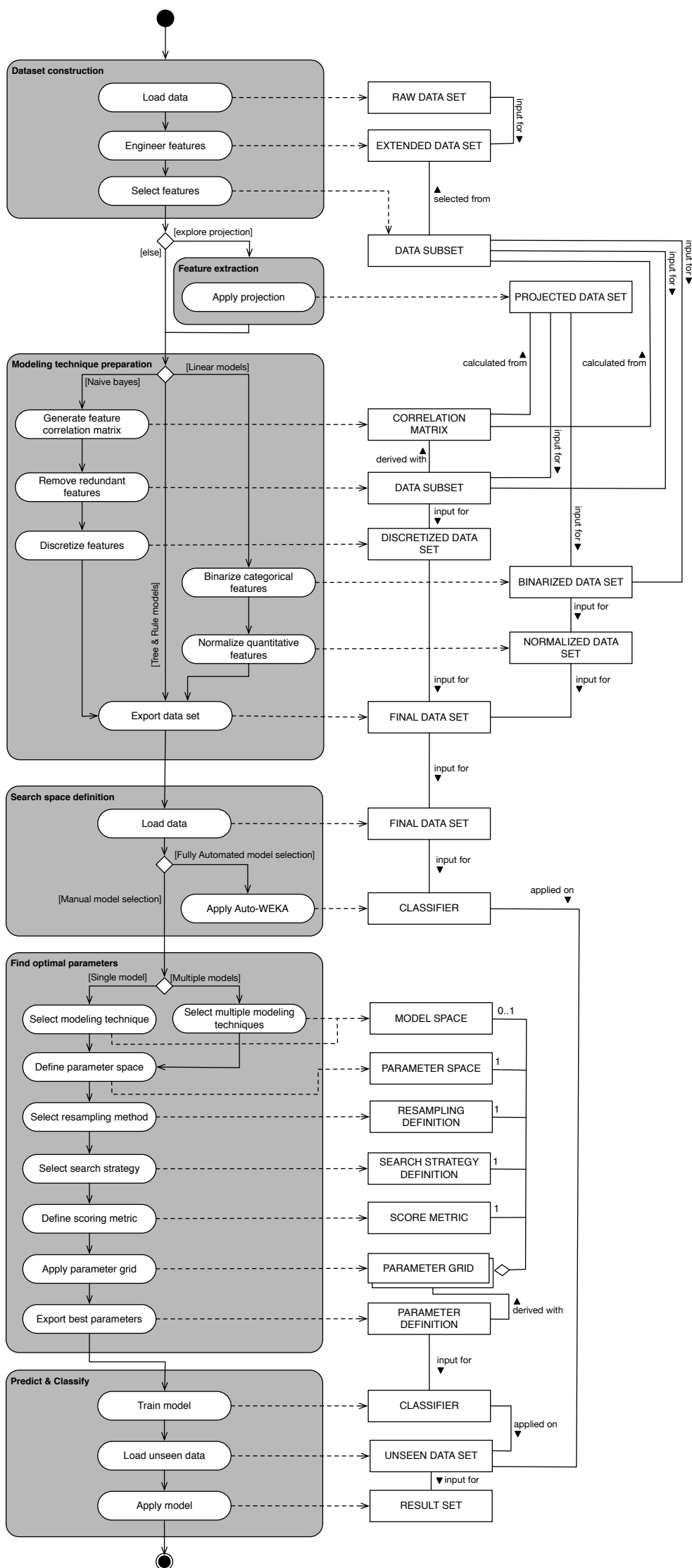
- Ho, Yu-Chi and David L. Pepyne (2002). "Simple explanation of the no-free-lunch theorem and its implications". In: *Journal of Optimization Theory and Applications* 115.3, pp. 549–570.
- Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin (2003). In: *A practical guide to support vector classification*.
- Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment". In: *Computing In Science & Engineering* 9.3, pp. 90–95.
- Jain, Anil K., M. Narasimha Murty, and Patrick J. Flynn (1999). "Data clustering: a review". In: *ACM computing surveys (CSUR)* 31.3, pp. 264–323.
- Jalali, Samireh and Claes Wohlin (2012). "Systematic literature studies: database searches vs. backward snowballing". In: *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*. ACM, pp. 29–38.
- Japkowicz, Nathalie and Mohak Shah (2015). "Performance Evaluation in Machine Learning". In: *Machine Learning in Radiation Oncology*. Springer, pp. 41–56.
- Johansson, Ulf, Lars Niklasson, and Rikard König (2004). "Accuracy vs. comprehensibility in data mining models". In: *Proceedings of the Seventh International Conference on Information Fusion*. Vol. 1. Citeseer, pp. 295–300.
- Kamwa, Innocent, SR Samantaray, and Geza Joós (2012a). "On the accuracy versus transparency trade-off of data-mining models for fast-response PMU-based catastrophe predictors". In: *Smart Grid, IEEE Transactions on* 3.1, pp. 152–161.
- (2012b). "On the accuracy versus transparency trade-off of data-mining models for fast-response PMU-based catastrophe predictors". In: *Smart Grid, IEEE Transactions on* 3.1, pp. 152–161.
- Karayiannis, Nicolaos and Anastasios N. Venetsanopoulos (2013). *Artificial neural networks: learning algorithms, performance evaluation, and applications*. Vol. 209. Springer Science & Business Media.
- Kay, Stanley R., Abraham Flszbein, and Lewis A. Opfer (1987). "The positive and negative syndrome scale (PANSS) for schizophrenia." In: *Schizophrenia bulletin* 13.2, p. 261.
- Kotsiantis, Sotiris B., I. Zaharakis, and P. Pintelas (2007). In: *Supervised machine learning: A review of classification techniques*.
- Kuhn, Max and Kjell Johnson (2013). *Applied predictive modeling*. Springer.
- Kumar, Rajeev and Abhaya Indrayan (2011). "Receiver operating characteristic (ROC) curve for medical researchers". In: *Indian pediatrics* 48.4, pp. 277–287.
- Lan, H., Eibe Frank, and MA Hall (2011). "Data mining: Practical machine learning tools and techniques". In:
- Lavrač, Nada (1999). "Selected techniques for data mining in medicine". In: *Artificial Intelligence in Medicine* 16.1, pp. 3–23.
- Locklin, Scott (2014). *Neglected machine learning ideas*.
- Lou, Yin, Rich Caruana, and Johannes Gehrke (2012). "Intelligible models for classification and regression". In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 150–158.
- Lou, Yin et al. (2013). "Accurate intelligible models with pairwise interactions". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 623–631.

- Mariscal, Gonzalo, Óscar Marbán, and Covadonga Fernández (2010a). "A survey of data mining and knowledge discovery process models and methodologies". In: *The Knowledge Engineering Review* 25.02, pp. 137–166.
- (2010b). "A survey of data mining and knowledge discovery process models and methodologies". In: *The Knowledge Engineering Review* 25.02, pp. 137–166.
- Markham, Kevin (2014). *Simple guide to confusion matrix terminology*.
- Martens, David et al. (2007). "Comprehensible credit scoring models using rule extraction from support vector machines". In: *European Journal of Operational Research* 183.3, pp. 1466–1476.
- Metzen, Jan Hendrik (2015). *Advice for applying machine learning*.
- Michalski, Ryszard S. (1975). "Synthesis of optimal and quasi-optimal variable-valued logic formulas". In:
- Milovic, Boris (2012). "Prediction and decision making in Health Care using Data Mining". In: *International Journal of Public Health Science (IJPHS)* 1.2, pp. 69–78.
- Mladenic, Dunja (2003). *Data mining and decision support: integration and collaboration*. Springer Science & Business Media.
- Murty, M. Narasimha and V. Susheela Devi (2011). "Support Vector Machines". In: *Pattern Recognition*. Springer, pp. 147–187.
- Myles, Anthony J. et al. (2004). "An introduction to decision tree modeling". In: *Journal of Chemometrics* 18.6, pp. 275–285.
- Olson, David L., Dursun Delen, and Yanyan Meng (2012). "Comparative analysis of data mining methods for bankruptcy prediction". In: *Decision Support Systems* 52.2, pp. 464–473.
- Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.
- Peng, Roger D. and Elizabeth Matsui (2016). *The Art of Data Science*. Leanpub.
- Phua, Clifton et al. (2010). "A comprehensive survey of data mining-based fraud detection research". In: *arXiv preprint arXiv:1009.6119*.
- Piatetsky, Gregory (2014). *CRISP-DM, still the top methodology for analytics, data mining, or data science projects*.
- Pressman, Roger S. (2005). *Software engineering: a practitioner's approach*. Palgrave Macmillan.
- Proudfoot, J. (2013). "The future is in our hands: the role of mobile phones in the prevention and management of mental disorders". eng. In: *The Australian and New Zealand Journal of Psychiatry* 47.2. LR: 20141120; JID: 0111052; ppublish, pp. 111–113.
- Quinlan, J. Ross (2014). *C4. 5: programs for machine learning*. Elsevier.
- Raschka, Sebastian (2014). *Implementing a Principal Component Analysis (PCA) in Python step by step*.
- Saeki, Motoshi (2003). "Embedding metrics into information systems development methods: An application of method engineering technique". In: *Advanced information systems engineering*. Springer, pp. 374–389.
- Scheper, WJ (2002). "Business IT Alignment: solution for the productivity paradox". In: *Deloitte & Touche, Netherlands*.
- Setiono, RUDY (2003). "Techniques for extracting classification and regression rules from artificial neural networks". In: *Computational intelligence: The experts speak*, pp. 99–114.

- Shumaker, BP and RW Sinnott (1984). "Astronomical computing: 1. Computing under the open sky. 2. Virtues of the haversine." In: *Sky and telescope* 68, pp. 158–159.
- Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams (2012). "Practical Bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems*, pp. 2951–2959.
- Sutton, Richard S. and Andrew G. Barto (1998). *Introduction to reinforcement learning*. Vol. 135. MIT Press Cambridge.
- Tang, Jiliang, Salem Alelyani, and Huan Liu (2014). "Feature selection for classification: A review". In: *Data Classification: Algorithms and Applications*, p. 37.
- Thornton, Chris et al. (2013). "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 847–855.
- Tukey, John W. (1977). "Exploratory data analysis". In: Verschuren, Piet, Hans Doorewaard, and MJ Mellion (2010). *Designing a research project*. Vol. 2. Eleven International Publishing.
- Vleugel, Arjen, Marco Spruit, and Anton van Daal (2009). "Historical data analysis through data mining from an outsourcing perspective: the Three-phases model". In:
- Waskom, Michael et al. (2015). *seaborn: v0.6.0 (June 2015)*. DOI: 10.5281/zenodo.19108. URL: <http://dx.doi.org/10.5281/zenodo.19108>.
- Weerd, Inge van de and Sjaak Brinkkemper (2008). "Meta-modeling for situational analysis and design methods". In: *Handbook of research on modern systems analysis and design technologies and applications* 35.
- Wohlin, Claes et al. (2012). *Experimentation in software engineering*. Springer Science & Business Media.
- Wold, Svante, Kim Esbensen, and Paul Geladi (1987). "Principal component analysis". In: *Chemometrics and Intelligent Laboratory Systems* 2.1, pp. 37–52.
- Yang, Ying and Geoffrey I. Webb (2003). "On Why Discretization Works for Naive-Bayes Classifiers". English. In: *AI 2003: Advances in Artificial Intelligence*. Ed. by Tamás(Tom)Domonkos Gedeon and LanceChunChe Fung. Vol. 2903. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 440–452. ISBN: 978-3-540-20646-0. DOI: 10.1007/978-3-540-24581-0_37. URL: http://dx.doi.org/10.1007/978-3-540-24581-0_37.
- Yoo, Illhoi et al. (2012). "Data mining in healthcare and biomedicine: a survey of the literature". In: *Journal of medical systems* 36.4, pp. 2431–2448.

Appendix A

The Integrated Method



Appendix B

Behapp data set

incoming_calls	outgoing_calls	missed_calls	call_duration	incoming_sms	outgoing_sms	incoming_mms	outgoing_mms	twitter_personal_tweets
1,59	1,52	0,98	9,02	0,82	0,26	0,10	0,02	0,01
0,31	0,74	0,09	2,66	0,39	0,15	0,10	0,00	0,00
0,20	0,45	0,21	0,63	0,63	0,18	0,00	0,00	0,59
0,06	0,10	0,02	0,11	0,14	0,05	0,00	0,00	0,02
2,38	1,08	0,38	15,07	4,00	1,31	0,00	0,00	0,00
0,08	0,20	0,08	0,46	0,06	0,00	0,04	0,00	0,00
1,09	0,45	0,09	14,15	0,82	0,55	0,00	0,00	0,00
0,21	1,29	0,71	6,48	0,71	0,36	0,00	0,00	0,00
1,30	4,17	1,47	5,42	4,63	3,83	1,27	0,00	0,00
0,47	0,44	0,08	4,46	0,81	0,53	0,00	0,39	0,00
0,13	0,84	0,15	2,65	0,55	0,17	0,47	0,34	0,11
1,41	1,47	0,88	4,64	0,76	0,18	0,47	0,00	0,00
0,88	1,15	0,15	6,90	0,55	0,33	0,00	0,00	0,00
1,33	2,53	0,07	10,22	1,27	0,80	0,00	0,00	0,00
0,79	1,36	0,64	4,26	0,21	0,00	0,57	0,00	0,00
0,68	0,96	0,36	19,29	0,68	0,56	0,68	0,00	0,00
0,75	1,44	0,63	1,75	1,56	0,00	0,00	0,00	0,00

twitter_direct_incoming	twitter_direct_outgoing	facebook_timeline_post	application_activity	whatsapp_activity	bluetooth_devices_detected
0,00	0,00	0,00	21,31	3,45	151,86
0,00	0,00	0,26	25,69	3,36	311,64
0,20	0,06	0,05	55,69	16,74	3,69
0,00	0,00	0,79	3,30	2,30	9,32
0,00	0,00	0,00	106,23	13,00	33,38
0,00	0,00	1,11	0,00	0,00	0,93
0,00	0,00	0,00	65,27	9,00	11,27
0,00	0,00	0,00	102,14	23,71	2,36
0,00	0,00	0,00	47,70	5,17	4,60
0,00	0,00	0,00	100,44	39,64	2,97
0,00	0,00	0,00	0,03	0,00	58,64
0,00	0,00	0,00	0,00	0,00	82,24
0,00	0,00	0,00	28,12	3,27	1,00
0,00	0,00	0,00	0,20	0,00	1,87
0,00	0,00	0,00	0,00	0,00	26,57
0,00	0,00	0,00	0,04	0,00	23,00
0,00	0,00	0,00	212,19	59,06	22,81

bluetooth_detection_events	bluetooth_device_diversity	bluetooth_detection_ratio	unique_contacts	places_visited_daily	places_visited_diversity
55,77	4.400,00	2,72	794,00	2,69	0,00
16,73	551,00	18,63	598,00	1,76	62,00
81,68	124,00	0,05	55,00	1,18	8,00
27,47	320,00	0,34	13,00	1,00	1,00
38,08	151,00	0,88	57,00	2,33	10,00
7,99	59,00	0,12	14,00	0,33	2,00
25,27	101,00	0,45	18,00	1,36	8,00
17,36	9,00	0,14	13,00	5,00	19,00
11,60	125,00	0,40	22,00	1,00	2,00
26,00	72,00	0,11	20,00	1,22	5,00
134,30	1.401,00	0,44	77,00	3,30	0,00
89,35	230,00	0,92	16,00	3,44	24,00
13,79	27,00	0,07	23,00	0,00	0,00
5,13	9,00	0,36	27,00	0,60	3,00
65,71	205,00	0,40	12,00	1,38	4,00
14,28	298,00	1,61	28,00	1,00	3,00
89,69	203,00	0,25	18,00	3,70	17,00

patient
0
0
0
0
0
0
0
0
0
1
0
1
0
1
1
0
0
0

Appendix C

Experiment notebook - Iris data

Experiment_Iris

May 15, 2016

1 Experiment notebook - Iris data

This notebook contains the experiment procedure and corresponding output data of the experiment as conducted in chapter 5. The experiment has been applied to the Iris and Behapp data set, this instance is applied to the Iris data set. `## Load base packages`

```
In [1]: %matplotlib inline
```

```
# Import base packages
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import datasets

# Import classifier packages
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier

# Import cross validation package
from sklearn.cross_validation import cross_val_score

# Import grid search (support) package(s)
from sklearn.grid_search import GridSearchCV
from operator import itemgetter
from time import time

# Metric definition ("roc_auc", "accuracy")
metric = "accuracy"

# Specify the number of folds
folds = 4
```

```
In [2]: # Utility function to report best scores for the grid searches
def report(grid_scores, n_top=5):
    top_scores = sorted(grid_scores, key=itemgetter(1), reverse=True)[:n_top]
    for i, score in enumerate(top_scores):
        print("Model with rank: {}".format(i + 1))
        print("Mean validation score: {:.3f} (std: {:.3f})".format(
            score.mean_validation_score,
            np.std(score.cv_validation_scores)))
        print("Parameters: {}".format(score.parameters))
    print("")
```

1.1 Load the Iris data set

```
In [3]: # Load the iris data set.
iris = datasets.load_iris()
x = pd.DataFrame(iris.data)
y = pd.DataFrame(iris.target)
y = y.rename(columns={0: 'type'})

# Remove the Setosa instances to convert the set to a two class data set
flowers = pd.concat([x,y], axis=1)
flowers = flowers[flowers.type != 0]

# Separate versicolor and virgina attribute information from the class information
flowers_x = flowers.ix[:,0:4].values
flowers_y = flowers.ix[:,4]
flowers_y = flowers_y.replace(to_replace=1, value=0)
flowers_y = flowers_y.replace(to_replace=2, value=1)
flowers_y = flowers_y.values
```

1.2 Plain method

1.2.1 Decision tree (Tree model)

```
In [4]: # Instantiate the decision tree classifier
clf = DecisionTreeClassifier()

# Run the test using 10 fold cross validation
scores = cross_val_score(clf, flowers_x, flowers_y, scoring=metric, cv=folds)
scores.mean()
```

Out[4]: 0.93990384615384615

1.2.2 Support vector machine (Linear model)

```
In [5]: # Instantiate the support vector machine classifier
clf = svm.SVC()

# Run the test using 10 fold cross validation
scores = cross_val_score(clf, flowers_x, flowers_y, scoring=metric, cv=folds)
scores.mean()
```

Out[5]: 0.95913461538461542

1.2.3 Random forest (Ensemble)

```
In [6]: # Instantiate the random forest classifier
clf = RandomForestClassifier()

# Run the test using 10 fold cross validation
scores = cross_val_score(clf, flowers_x, flowers_y, scoring=metric, cv=folds)
scores.mean()
```

Out[6]: 0.92948717948717952

1.3 Integrated method

1.3.1 Feature selection

We apply univariate feature selection as part of the data preparation phase in the integrated approach. The two most informative features are selected.

```
In [7]: from sklearn.feature_selection import SelectKBest
        from sklearn.feature_selection import chi2

        flowers_x_selected = SelectKBest(chi2, k=2).fit_transform(flowers_x, flowers_y)
```

1.3.2 Decision tree (Tree model) - Using grid search with 8 candidate parameter settings

```
In [8]: # build a classifier
        clf = DecisionTreeClassifier()

        # use a full grid over all parameters
        param_grid = {"max_features": [1, 2],
                      "splitter": ["best", "random"],
                      "criterion": ["gini", "entropy"]}

        # run grid search
        grid_search = GridSearchCV(clf, param_grid=param_grid, scoring=metric, cv=folds)
        start = time()
        grid_search.fit(flowers_x_selected, flowers_y)

        print("GridSearchCV took %.2f seconds for %d candidate parameter settings."
              % (time() - start, len(grid_search.grid_scores_)))
        report(grid_search.grid_scores_)
```

GridSearchCV took 0.06 seconds for 8 candidate parameter settings.

Model with rank: 1

Mean validation score: 0.940 (std: 0.020)

Parameters: {'max_features': 1, 'splitter': 'random', 'criterion': 'gini'}

Model with rank: 2

Mean validation score: 0.930 (std: 0.035)

Parameters: {'max_features': 2, 'splitter': 'best', 'criterion': 'gini'}

Model with rank: 3

Mean validation score: 0.930 (std: 0.035)

Parameters: {'max_features': 2, 'splitter': 'random', 'criterion': 'gini'}

Model with rank: 4

Mean validation score: 0.930 (std: 0.055)

Parameters: {'max_features': 1, 'splitter': 'random', 'criterion': 'entropy'}

Model with rank: 5

Mean validation score: 0.930 (std: 0.035)

Parameters: {'max_features': 2, 'splitter': 'best', 'criterion': 'entropy'}

1.3.3 Random forest (Ensemble) - Using grid search with 80 candidate parameter settings

```
In [9]: # build a classifier
        clf = RandomForestClassifier()
```

```

# use a full grid over all parameters
param_grid = {"max_features": [1, 2],
              "n_estimators": [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],
              "criterion": ["gini", "entropy"]}

# run grid search
grid_search = GridSearchCV(clf, param_grid=param_grid, scoring=metric, cv=folds)
start = time()
grid_search.fit(flowers_x_selected, flowers_y)

print("GridSearchCV took %.2f seconds for %d candidate parameter settings."
      % (time() - start, len(grid_search.grid_scores_)))
report(grid_search.grid_scores_)

```

GridSearchCV took 3.42 seconds for 80 candidate parameter settings.

Model with rank: 1

Mean validation score: 0.950 (std: 0.033)

Parameters: {'max_features': 1, 'n_estimators': 12, 'criterion': 'gini'}

Model with rank: 2

Mean validation score: 0.950 (std: 0.033)

Parameters: {'max_features': 1, 'n_estimators': 14, 'criterion': 'gini'}

Model with rank: 3

Mean validation score: 0.950 (std: 0.033)

Parameters: {'max_features': 1, 'n_estimators': 15, 'criterion': 'gini'}

Model with rank: 4

Mean validation score: 0.950 (std: 0.033)

Parameters: {'max_features': 1, 'n_estimators': 17, 'criterion': 'gini'}

Model with rank: 5

Mean validation score: 0.950 (std: 0.033)

Parameters: {'max_features': 2, 'n_estimators': 6, 'criterion': 'gini'}

1.3.4 Data standardization

As part of the data preparation phase we standardize the data before feeding it to the support vector machine (linear model).

In [10]: `from sklearn import preprocessing`

```

# Compute the mean and std for all columns
flowers_scale_columns = preprocessing.StandardScaler().fit(flowers_x_selected)

# Standardize columns (centering & scaling)
flowers_x_selected_scaled = flowers_scale_columns.transform(flowers_x_selected)

```

1.3.5 Support vector machine (Linear model) - Grid search with 144 candidate parameter settings

In [11]: `# build a classifier`
`clf = svm.SVC()`

```

# use a full grid over all parameters
param_grid = {"C": np.logspace(-3, 2, 6),
              "gamma": np.logspace(-3, 2, 6),
              "kernel": ["linear", "poly", "rbf", "sigmoid"]}

# run grid search
grid_search = GridSearchCV(clf, param_grid=param_grid, scoring=metric, cv=folds)
start = time()
grid_search.fit(flowers_x_selected_scaled, flowers_y)

print("GridSearchCV took %.2f seconds for %d candidate parameter settings."
      % (time() - start, len(grid_search.grid_scores_)))
report(grid_search.grid_scores_)

```

GridSearchCV took 4.73 seconds for 144 candidate parameter settings.

Model with rank: 1

Mean validation score: 0.960 (std: 0.051)

Parameters: {'kernel': 'poly', 'C': 0.001, 'gamma': 100.0}

Model with rank: 2

Mean validation score: 0.960 (std: 0.051)

Parameters: {'kernel': 'poly', 'C': 1.0, 'gamma': 10.0}

Model with rank: 3

Mean validation score: 0.950 (std: 0.033)

Parameters: {'kernel': 'rbf', 'C': 0.001, 'gamma': 0.10000000000000001}

Model with rank: 4

Mean validation score: 0.950 (std: 0.033)

Parameters: {'kernel': 'sigmoid', 'C': 0.001, 'gamma': 0.10000000000000001}

Model with rank: 5

Mean validation score: 0.950 (std: 0.033)

Parameters: {'kernel': 'linear', 'C': 0.01, 'gamma': 0.001}

Appendix D

Experiment notebook - Behapp data

Experiment_behapp

May 15, 2016

1 Experiment notebook - Behapp data

This notebook contains the experiment procedure and corresponding output data of the experiment as conducted in chapter 5. The experiment has been applied to the Iris and Behapp data set, this instance is applied to the Behapp data set.

1.1 Load base packages

```
In [1]: %matplotlib inline
```

```
# Import base packages
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import datasets

# Import classifier packages
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier

# Import cross validation package
from sklearn.cross_validation import cross_val_score

# Import grid search (support) package(s)
from sklearn.grid_search import GridSearchCV
from operator import itemgetter
from time import time

# Metric definition ("roc_auc", "accuracy")
metric = "accuracy"

# Specify the number of folds
folds = 4
```

```
In [2]: # Utility function to report best scores for the grid searches
```

```
def report(grid_scores, n_top=5):
    top_scores = sorted(grid_scores, key=itemgetter(1), reverse=True)[:n_top]
    for i, score in enumerate(top_scores):
        print("Model with rank: {0}".format(i + 1))
        print("Mean validation score: {0:.3f} (std: {1:.3f})".format(
            score.mean_validation_score,
```

```

        np.std(score.cv_validation_scores)))
print("Parameters: {0}".format(score.parameters))
print("")

```

1.2 Load the Behapp data set

```

In [3]: # Load the data set into memory
        behapp = pd.read_csv('overview_std_dly.csv')

        # Drop the person id and class column and store the rest in the the X variable
        behapp_x = behapp.ix[:,1:22].values

        # Select the class column and store this in the y variable
        behapp_y = behapp.ix[:,22].values

```

1.3 Plain method

1.3.1 Decision tree (Tree model)

```

In [4]: # Instantiate the decision tree classifier
        clf = DecisionTreeClassifier()

        # Run the test using 10 fold cross validation
        scores = cross_val_score(clf, behapp_x, behapp_y, scoring=metric, cv=folds)
        scores.mean()

```

Out[4]: 0.5999999999999998

1.3.2 Support vector machine (Linear model)

```

In [5]: # Instantiate the support vector machine classifier
        clf = svm.SVC()

        # Run the test using 10 fold cross validation
        scores = cross_val_score(clf, behapp_x, behapp_y, scoring=metric, cv=folds)
        scores.mean()

```

Out[5]: 0.7624999999999996

1.3.3 Random forest (Ensemble)

```

In [6]: # Instantiate the random forest classifier
        clf = RandomForestClassifier()

        # Run the test using 10 fold cross validation
        scores = cross_val_score(clf, behapp_x, behapp_y, scoring=metric, cv=folds)
        scores.mean()

```

Out[6]: 0.7624999999999996

1.4 Integrated method

1.4.1 Feature selection

We apply univariate feature selection as part of the data preparation phase in the integrated approach. The two most informative features are selected.

```
In [7]: from sklearn.feature_selection import SelectKBest
        from sklearn.feature_selection import chi2

        behapp_x_selected = SelectKBest(chi2, k=2).fit_transform(behapp_x, behapp_y)
```

1.4.2 Decision tree (Tree model) - Using grid search with 8 candidate parameter settings

```
In [8]: # build a classifier
        clf = DecisionTreeClassifier()

        # use a full grid over all parameters
        param_grid = {"max_features": [1, 2],
                      "splitter": ["best", "random"],
                      "criterion": ["gini", "entropy"]}

        # run grid search
        grid_search = GridSearchCV(clf, param_grid=param_grid, scoring=metric, cv=folds)
        start = time()
        grid_search.fit(behapp_x_selected, behapp_y)

        optimal_tree = grid_search.best_estimator_

        print("GridSearchCV took %.2f seconds for %d candidate parameter settings."
              % (time() - start, len(grid_search.grid_scores_)))
        report(grid_search.grid_scores_)
```

GridSearchCV took 0.15 seconds for 8 candidate parameter settings.

Model with rank: 1

Mean validation score: 0.765 (std: 0.144)

Parameters: {'max_features': 2, 'splitter': 'best', 'criterion': 'entropy'}

Model with rank: 2

Mean validation score: 0.706 (std: 0.065)

Parameters: {'max_features': 1, 'splitter': 'best', 'criterion': 'gini'}

Model with rank: 3

Mean validation score: 0.706 (std: 0.065)

Parameters: {'max_features': 1, 'splitter': 'random', 'criterion': 'gini'}

Model with rank: 4

Mean validation score: 0.706 (std: 0.065)

Parameters: {'max_features': 2, 'splitter': 'best', 'criterion': 'gini'}

Model with rank: 5

Mean validation score: 0.706 (std: 0.188)

Parameters: {'max_features': 2, 'splitter': 'random', 'criterion': 'entropy'}

1.4.3 Random forest (Ensemble) - Using grid search with 80 candidate parameter settings

```
In [9]: # build a classifier
        clf = RandomForestClassifier()

        # use a full grid over all parameters
        param_grid = {"max_features": [1, 2],
                      "n_estimators": [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],
```

```

        "criterion": ["gini", "entropy"]}]

# run grid search
grid_search = GridSearchCV(clf, param_grid=param_grid, scoring=metric, cv=folds)
start = time()
grid_search.fit(behapp_x_selected, behapp_y)

print("GridSearchCV took %.2f seconds for %d candidate parameter settings."
      % (time() - start, len(grid_search.grid_scores_)))
report(grid_search.grid_scores_)

```

GridSearchCV took 9.44 seconds for 80 candidate parameter settings.

Model with rank: 1

Mean validation score: 0.882 (std: 0.114)

Parameters: {'max_features': 1, 'n_estimators': 9, 'criterion': 'gini'}

Model with rank: 2

Mean validation score: 0.882 (std: 0.114)

Parameters: {'max_features': 2, 'n_estimators': 14, 'criterion': 'gini'}

Model with rank: 3

Mean validation score: 0.882 (std: 0.114)

Parameters: {'max_features': 1, 'n_estimators': 12, 'criterion': 'entropy'}

Model with rank: 4

Mean validation score: 0.882 (std: 0.114)

Parameters: {'max_features': 1, 'n_estimators': 14, 'criterion': 'entropy'}

Model with rank: 5

Mean validation score: 0.882 (std: 0.114)

Parameters: {'max_features': 1, 'n_estimators': 17, 'criterion': 'entropy'}

1.4.4 Data standardization

As part of the data preparation phase we standardize the data before feeding it to the support vector machine (linear model).

In [10]: `from sklearn import preprocessing`

```

# Compute the mean and std for all columns
behapp_scale_columns = preprocessing.StandardScaler().fit(behapp_x_selected)

# Standardize columns (centering & scaling)
behapp_x_selected_scaled = behapp_scale_columns.transform(behapp_x_selected)

```

1.4.5 Support vector machine (Linear model) - Grid search with 144 candidate parameter settings

In [11]: `# build a classifier`

```

clf = svm.SVC()

# use a full grid over all parameters
param_grid = {"C": np.logspace(-3, 2, 6),
              "gamma": np.logspace(-3, 2, 6),
              "kernel": ["linear", "poly", "rbf", "sigmoid"]}

```

```

    }

    # run grid search
    grid_search = GridSearchCV(clf, param_grid=param_grid, scoring=metric, cv=folds)
    start = time()
    grid_search.fit(behapp_x_selected_scaled, behapp_y)

    print("GridSearchCV took %.2f seconds for %d candidate parameter settings."
          % (time() - start, len(grid_search.grid_scores_)))
    report(grid_search.grid_scores_)

GridSearchCV took 12.75 seconds for 144 candidate parameter settings.
Model with rank: 1
Mean validation score: 0.824 (std: 0.205)
Parameters: {'kernel': 'rbf', 'C': 10.0, 'gamma': 100.0}

Model with rank: 2
Mean validation score: 0.824 (std: 0.103)
Parameters: {'kernel': 'rbf', 'C': 100.0, 'gamma': 100.0}

Model with rank: 3
Mean validation score: 0.765 (std: 0.022)
Parameters: {'kernel': 'linear', 'C': 0.001, 'gamma': 0.001}

Model with rank: 4
Mean validation score: 0.765 (std: 0.022)
Parameters: {'kernel': 'poly', 'C': 0.001, 'gamma': 0.001}

Model with rank: 5
Mean validation score: 0.765 (std: 0.022)
Parameters: {'kernel': 'rbf', 'C': 0.001, 'gamma': 0.001}

In [12]: from sklearn import tree
         tree.export_graphviz(optimal_tree, out_file='behapp.dot',
                             feature_names=["bluetooth_device_diversity", "unique_contacts"],
                             class_names=["control", "patients"],
                             filled=True, rounded=True,
                             special_characters=True
                             )

In [13]: import sklearn
         print sklearn.__version__

0.17

```

Appendix E

Notebook - Behapp data wrangling

Behapp data wrangling

May 15, 2016

1 Behapp project - Data wrangling

This notebook describes and reports on the data acquired as part of the behapp project. The project entails a long term study of patients suffering from mental disorders and a control group. Participants were required to run a passive mobile monitoring app which recorded various social and movements 'acts' (e.g. incoming phone call and gps coordinates). The goal of this analysis is to determine whether features exist that separate patients from control group participants.

We will run through this analysis on a step by step basis covering three aspects from the CRISP-DM process model: data understanding, data preparation and modeling. Since the time series data of the study requires transformation to single participant records the understanding phase will be part of the preparation and modeling phases.

```
In [1]: # First we import all modules and packages that will be needed for the analysis.
        %matplotlib inline
        import numpy as np
        import pandas as pd
        import seaborn as sns
        from scipy import stats
        import matplotlib as mpl
        import matplotlib.pyplot as plt
        from sklearn.cluster import DBSCAN
        from sklearn import preprocessing

        # Radius and distance parameters for the clustering algorithm.
        min_samples = 20
        distance = 150

        # The minimal number of days a participant should participate in the study for data analysis.
        minimum_days = 1
```

1.1 1. Data preparation

1.1.1 1.1 Data import/integration

We start by consolidating all datasources and merging them in a denormalized dataset. First we open all the datafiles that have been exported as CSV (comma separated values) files.

```
In [2]: # Import the (openclinica) participant data from the behapp study export.
        oc_participant_data = pd.read_excel('sep_2015/oc_behapp.xlsx')

        # Import the participant data from the behapp datase.
        behapp_participant_data = pd.read_csv('sep_2015/person.csv')

        # Import the basepoints of the participants in the behapp study.
```



```

behapp_participant_basepoints = pd.read_csv('sep_2015/basepoint.csv')

# Import the event names and their corresponding ID's from the behapp database.
behapp_event_names = pd.read_csv('sep_2015/event_type.csv')

# Import the recorded event data of the behapp participants.
behapp_events = pd.read_csv('sep_2015/message.csv')

```

1.1.2 1.2 Data merge

1.2.1 Merge behapp-participant table with the openclinica-participant table As a security measure, patient status and further personal details are not stored in the behapp database. These details are kept in a more secure internal clinical study registration system (openclinica). However for our analysis purposes we require the patient status to be available in the time series events overview. Therefore we start by merging the openclinica data with the behapp participant data. They share a common column 'userid' which is a small hash.

```

In [3]: # First we select the needed columns from both participant datasets.
oc_uid = oc_participant_data.loc[:,[' PANSS_E1_C1 ', ' app_code_E1_C1 ']]
behapp_uid = behapp_participant_data.loc[:,['uid', 'id']]

# Join the openclinica and the behapp data on their common columns in order to match the patient.
behapp_participant_overview = pd.merge(oc_uid,
                                       behapp_uid,
                                       left_on=' app_code_E1_C1 ',
                                       right_on='uid',
                                       how='inner')

# Rename the PANSS_E1_C1 column to patient.
behapp_participant_overview = behapp_participant_overview.rename(columns={' PANSS_E1_C1 ':'patient'})

# Replace all NaN values with the value 0 (zero).
behapp_participant_overview = behapp_participant_overview.fillna(value='0')

# Remove the redundant app_code_E1_C1 and uid columns and finalize participant overview.
behapp_participant_overview = behapp_participant_overview.loc[:,['id', 'patient']]

```

1.2.2 Merge participant overview with events overview The events overview is a table which contains the recorded events of all participants involved in the behapp study. Events are actions like incoming phone calls and movements that are recorded by the behapp mobile monitoring app. Each event has a unique event.id and various properties like the date and time and ofcourse the corresponding person.id. Furthermore when relevant additional properties are recorded like GPS coordinates and / or in the case of incoming phone calls an encrypted hash of the contact of the participant.

We continue denormalizing the events overview by merging the participant overview with the events overview.

```

In [4]: # Merge the behapp dataset with the participant overview.
behapp_events = pd.merge(behapp_events,
                          behapp_participant_overview,
                          left_on='person_id',
                          right_on='id',
                          how='inner')

```

1.2.3 Merge basepoint table with events overview Next we merge the basepoint table with the events overview. The basepoint table contains coordinates that are marked as 'homebases' or points that are frequently visited by participants.

```
In [5]: # Prepare the basepoint table for merging with the events overview.
        behapp_participant_basepoints = behapp_participant_basepoints.loc[:, ['id', 'name']]

        # Merge the basepoints with the event data.
        behapp_events = pd.merge(behapp_events,
                                behapp_participant_basepoints,
                                left_on='basepoint_id',
                                right_on='id',
                                how='outer')
```

1.2.4 Merge event names with the events overview Last to make it easier to understand what each event_id means we merge in the event names of each event_id from the event names table.

```
In [6]: # Merge the event names with the event data.
        behapp_events = pd.merge(behapp_events,
                                behapp_event_names,
                                left_on='event_id',
                                right_on='id',
                                how='inner')
```

1.1.3 1.3 Data clean

To wrap it all up we rename a couple of columns for better understanding. Next we select the columns that we want to use, at this point we choose to drop the columns speed and duration since they are not found to be of use to the analysis. Next we perform various actions related to indexing setting of the dataset, these settings are related to the package used to transform this data (pandas). They allow us to slice (select) the data intelligently which we will extensively use in building the consolidated overview. Last we extract person_id's and event_id's from the events overview self ensuring that we run our analysis from a single source of truth.

```
In [7]: # Rename column names for better clarity.
        behapp_events = behapp_events.rename(columns ={'name': 'basepoint_name',
                                                    'type': 'event_name',
                                                    'time': 'date_time'})

        # Select the columns to be used in the final dataset.
        behapp_events = behapp_events.loc[:, ['person_id',
                                              'event_id',
                                              'date_time',
                                              'lat', 'lng',
                                              'basepoint_id',
                                              'basepoint_name',
                                              'event_name',
                                              'receiver',
                                              'quantity',
                                              'duration',
                                              'speed',
                                              'patient']]
```

Filter leisure time In this intermediary section we pass a function to filter the leasure time (currently disabled).

```
In [8]: #behapp_events['date_time'] = pd.to_datetime(behapp_events['date_time'])
        #behapp_events = behapp_events.set_index(['date_time'], drop=False)

        #hours = behapp_events.index.hour
        #days = behapp_events.index.weekday
        #selector = ((days <= 4) & (hours < 19))
        ## Inverse mask application with ~
        #behapp_events = behapp_events[~selector]
        #behapp_events = behapp_events.reset_index(drop=True)
```

Next we continue with the normal analysis.

```
In [9]: # Set the index on person_id and event_id to support dataframe slicing (1/3).
        behapp_events = behapp_events.set_index(['person_id', 'event_id'], drop=False)

        # Lexically sort all the data to support dataframe slicing (2/3).
        behapp_events = behapp_events.sortlevel(0, sort_remaining=True)

        # Instantiate the IndexSlice method (3/3).
        idx = pd.IndexSlice

        # Replace all NaN values with the value 0 (zero).
        behapp_events = behapp_events.fillna(value='0')

        # Convert the date_time column from string to datetime (neccessary for date calculations later
        behapp_events['date_time'] = pd.to_datetime(behapp_events['date_time'])

        # Extract the unique participant id's and the (used) event_id's from the event dataframe.
        participants_srs = pd.Series(pd.unique(behapp_events.person_id))
        events_srs = pd.Series(pd.unique(behapp_events.event_id))

        # Set the proper datatypes on the columns of the dataset
        behapp_events['person_id'] = behapp_events['person_id'].astype(int)
        behapp_events['event_id'] = behapp_events['event_id'].astype(int)
        behapp_events['lat'] = behapp_events['lat'].astype(float)
        behapp_events['lng'] = behapp_events['lng'].astype(float)
        behapp_events['basepoint_id'] = behapp_events['basepoint_id'].astype(int)
        behapp_events['quantity'] = behapp_events['quantity'].astype(float)
        behapp_events['duration'] = behapp_events['duration'].astype(float)
        behapp_events['speed'] = behapp_events['speed'].astype(float)
        behapp_events['patient'] = behapp_events['patient'].astype(int)

        # Save a copy of the data to file
        behapp_events.to_csv('sep_2015/cleansed_events.csv', index=False)
```

1.1.4 1.4 Create overview table

1.4.1 Basic counts of total number of events per participant In this section we will start by instantiating a new dataframe that will hold all the participants and their features that will be used during the modeling phase. The dataframe called 'overview' will initially hold counts of the number of events that are registered for each participant.

```
In [10]: overview = pd.DataFrame()
        overview.insert(0, 'person_id', participants_srs)
        overview = overview.set_index('person_id', drop=True)
```

```

# Double for loop to determine the number of events for each participant per event_type.
for participant in participants_srs.iteritems():
    for event in events_srs.iteritems():
        if event[1] in behapp_events.loc[idx[participant[1],:],idx['event_id']].values:
            events_holder = behapp_events.loc[idx[participant[1],event[1]],:]
            overview.set_value(participant[1],event[1],len(events_holder))
        else:
            overview.set_value(participant[1],event[1],0)

# Function to replace the columns names to show the real names instead of the event_type integ
for event in events_srs.iteritems():
    event_name = behapp_events.loc[idx[:,event[1]],idx['event_name']].unique()
    overview.rename(columns={event[1]:event_name[0].strip()},inplace=True)

/Users/raj/anaconda/lib/python2.7/site-packages/pandas/core/indexing.py:1034: FutureWarning: scalar ind
obj = self._convert_scalar_indexer(obj, axis)
/Users/raj/anaconda/lib/python2.7/site-packages/pandas/core/indexing.py:1034: FutureWarning: scalar ind
obj = self._convert_scalar_indexer(obj, axis)

```

1.4.2 Number of events per type Below we report on the total number of events per type. We notice that the majority of events are of the passive kind were the app registers positions, detection of other devices and application activity. The active kind of events were users are interacting with the world show considerably lower counts.

```

In [11]: events_count = pd.DataFrame(overview.sum(axis=0))
events_count.rename(columns={0:'count'}, inplace=True)
events_count = events_count.sort_values(['count'], ascending=False)
events_count.to_excel('events_count.xls')
events_count

```

```

Out[11]:

```

	count
Position	377941
Number of detected Bluetooth devices	86984
Application activity	49258
Names and MAC addresses of detected Bluetooth d...	33360
Whats App outgoing message	1972
Outgoing call	1786
Incoming call	1370
Whats App incoming message	1266
Incoming SMS	1242
Proximity count	1143
Missed call	816
Outgoing SMS	502
Facebook timeline post	266
Incoming MMS	237
Twitter personal tweets	163
Outgoing MMS	71
Twitter direct incoming	50
Twitter direct outgoing	13

1.1.5 1.5 Extract features

In this section we extract / create features from the data that go beyond counting the occurrences of a certain event type.

1.5.1 Call duration

```
In [12]: for participant in participants_srs.iteritems():
         events_holder = behapp_events.loc[idx[participant[1],[4,5]],:]
         overview.set_value(participant[1], 'call_duration', events_holder.duration.sum()/60)
```

1.5.2 Whatsapp activity Whatsapp is one of the most popular messaging services in the Netherlands. Due to increasing safety measures the behapp initiative lost the ability to monitor the in-app activity of whatsapp. However we can still determine how often users use whatsapp.

```
In [13]: whatsapp_holder = behapp_events[behapp_events.receiver.str.contains("whatsapp")]

         for participant in participants_srs.iteritems():
             events_holder = whatsapp_holder.loc[idx[participant[1],:],:]
             overview.set_value(participant[1], 'whatsapp_activity', len(events_holder))
```

1.5.3 Determine size of social circle The mobile monitoring app registers contact information of the persons that have been in contact with the participants. This information is stored as an encrypted hash and serves the goal of determining unique values as to determine the size of a participants social circle.

Standardization over days will not be necessary for this feature.

```
In [14]: contacts = behapp_events[behapp_events.receiver.str.match('^[a-z0-9]{32}$').str.len() > 0]

         for participant in participants_srs.iteritems():
             events_holder = contacts.loc[idx[participant[1],:], idx['receiver']]
             events_holder = events_holder.unique()
             overview.set_value(participant[1], 'unique_contacts', len(events_holder))
```

```
/Users/raj/anaconda/lib/python2.7/site-packages/ipykernel/_main_.py:1: FutureWarning: In future versions
if __name__ == '__main__':
```

1.5.4 Detected bluetooth devices The app is able to monitor and detect other bluetooth-able devices in proximity. We sum the number of devices that have been detected each time when event_type 28 (Number of detected Bluetooth devices) has been registered by a device.

```
In [15]: for participant in participants_srs.iteritems():
         if 28 in behapp_events.loc[idx[participant[1],:], idx['event_id']].values:
             events_holder = behapp_events.loc[idx[participant[1],28],:]
             overview.set_value(participant[1], 'bluetooth_devices_detected', events_holder['quantity'])
```

1.5.5 Bluetooth device diversity We measure the different number of bluetooth devices that have been detected over time.

```
In [16]: for participant in participants_srs.iteritems():
         if 33 in behapp_events.loc[idx[participant[1],:], idx['event_id']].values:
             events_holder = behapp_events.loc[idx[participant[1],33],:]
             overview.set_value(participant[1], 'bluetooth_device_diversity', len(events_holder.receiver.unique()))
```

1.5.6 Bluetooth detection ratio

```
In [17]: overview['bluetooth_detection_ratio'] = overview['bluetooth_devices_detected'] / overview['Number_of_detected_Bluetooth_devices']
```

1.5.7 Determine daily map clusters Next to the number 2 (Position) event_type, almost all event types are registered with a set of gps (lat/lon) coordinates. These coordinates by itself are not immediately usable as a measure and preprocessing is therefore needed.

We propose to use an unsupervised density clustering algorithm (DBSCAN) which will be fed a daily overview of gps coordinates of each participant. The algorithm determines the clusters found in the daily overviews. Our reason of thinking is that each cluster represents a notable location and thus a ‘movement act’ of the participant. We sum the total number of clusters over all the days so that they can be standardized like the other features based on the number of days a participant has been involved with the study.

In this case we provide another days count column called ‘places_visited_days_count’ because it was found that the mobile monitoring app regularly failed to submit correct lists of coordinates. Instead the lists would only contain coordinates with values ‘0’. These values would be wrongfully classified as a cluster by the cluster algorithm.

Therefore the total number of days of correct GPS data could be lower than the earlier calculated number of days involved with the monitoring study (column: days).

In [18]: *# The following method determines the number of clusters based on a daily overview of gps (lat, lon) coordinates. These coordinates are passed as a numpy array. The clusters represent to what extent a participant moves. The method returns the total number of clusters minus the ‘noise’ cluster.*

```
def cluster_calc(day, participant_name, day_name='places_diversity'):

    global distance
    global min_samples

    # Select the lat and lng columns from the day variable.
    day_latlng = day[['lat', 'lng']].values

    # Convert the lat/lon values to radians as needed by the haversine distance metric
    day_radian = day_latlng * np.pi / 180.

    # Fit the DBSCAN density clustering algorithm to the lat/lon data array
    db = DBSCAN(eps=float(distance)/1000.0/6372.8,
                min_samples=min_samples,
                metric='haversine',
                algorithm='ball_tree').fit(day_radian)

    # Get a list of all the clustering results
    labels = db.labels_

    # Number of clusters in labels, ignoring noise if present.
    n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)

    # Create a new array to join the cluster labels with the original lat/lon coordinates.
    day_data = day.values
    clusterlabels = labels.reshape(1, -1).T
    coordinate_labels = pd.DataFrame(np.concatenate((day_data, clusterlabels), axis=1))
    coordinate_labels = coordinate_labels.rename(columns={0: 'date_time',
                                                         1: 'latitude',
                                                         2: 'longitude',
                                                         3: 'cluster'})

    # Store the daily GPS data in separate CSV files
    coordinate_labels.to_csv('clusterdata/' + str(participant_name) + '_' + str(day_name) + '.csv',
                           index=False)
```

```

        return n_clusters_

In [19]: # This function uses the cluster_calc function to determine the total number of daily clusters
         # for each participant and update the overview dataframe accordingly.

         for participant in participants_srs.iteritems():

             # Select the lat/lng and date_time columns for further analysis.
             events_holder = behapp_events.loc[idx[participant[1],:],idx['date_time','lat','lng']]

             # We reset the index in order to set the date_time column as the new index.
             events_holder = events_holder.reset_index(drop=True)

             # Convert the date_time column from the generic object datatype to the datetime datatype.
             events_holder['date_time'] = pd.to_datetime(events_holder['date_time'])

             # Set the index on the date_time column in order to allow for daily grouping of data.
             events_holder = events_holder.set_index(['date_time'], drop=False).groupby(pd.TimeGrouper(

             # Array that will temporarily hold the daily cluster data of each participant
             cluster_arr = []

             # Loop to walk through every day of gps data of the selected participant.
             for day in events_holder.__iter__():

                 global min_samples

                 # Pass the daily coordinates to a variable
                 daily_coords = day[1]

                 # Remove all the rows containing bogus coordinates (0.0)
                 daily_coords = daily_coords.query('lat != 0 & lng != 0')

                 # Remove all NaN values from the coordinates
                 daily_coords = daily_coords[np.isfinite(daily_coords['lat'])]
                 daily_coords = daily_coords[np.isfinite(daily_coords['lng'])]

                 # If a day contains holes and thus no gps data is present, skip it.
                 if len(daily_coords) >= min_samples:
                     # Save the estimated clusters and append to the cluster_arr array.
                     a = cluster_calc(daily_coords,participant[1],day[0])
                     cluster_arr.append(a)

             # Sum the daily cluster data and save the results for the participant in the overview data.
             overview.set_value(participant[1],'places_visited_daily',sum(cluster_arr))

             # Set a column with a count for the total number of days of GPS data.
             overview.set_value(participant[1],'places_visited_days_count',len(cluster_arr))

```

1.5.8 Determine map clusters (all gps data) We are also interested in the diversity of different locations that are visited by each participant. We therefore apply the clustering algorithm to the complete set of gps data.

```

In [20]: for participant in participants_srs.iteritems():
        if participant[1] != 13 and participant[1] != 105:

            # Select the lat/lng and date_time columns for further analysis.
            events_holder = behapp_events.loc[idx[participant[1],:],idx['date_time','lat','lng'],'srs']
            events_holder = events_holder.query('lat != 0 & lng != 0').query('speed == 0')
            events_holder = events_holder.loc[:,idx['date_time','lat','lng']]

            global min_samples

            if len(events_holder) >= min_samples:
                number_of_places = cluster_calc(events_holder,participant[1])
                overview.set_value(participant[1],'places_visited_diversity',number_of_places)

```

1.5.9 Active number of days The participants involved in the study are instructed to participate for at least two weeks with a preference for a longer period. In practice this has lead to wildly varying period of activity for each participant. In order to standardize the feature measurement to a common denominator (days) we need to determine the number of days a participant has been actively involved with the study. We calculate this number in two ways:

1- The number_of_days recorded represents all of the 'unique' days that are found for each corresponding participant in the events overview. This way of counting overcomes 'holes' in the data since the app sometimes fails to register data for several days in a row.

```

In [21]: for participant in participants_srs.iteritems():
        events_holder = behapp_events.loc[idx[participant[1],:],idx['date_time']].map(lambda t: t.date)
        overview.set_value(participant[1],'number_of_unique_days',len(events_holder))

```

2- The number_of_days.registered represents the exact time between the first and last registered event of each participant. In the case that the data does not contain any holes this number will usually be lower than the first count. For example, it is entirely possible that only 13 days and 12 hours can pass during a 15 day period. We will therefore prefer to use this count to standardize all the features as it is more precise.

```

In [22]: for participant in participants_srs.iteritems():
        events_holder = behapp_events.loc[idx[participant[1],:],idx['date_time']]
        date_diff = events_holder.max() - events_holder.min()
        days = pd.Timedelta(date_diff).days+1
        overview.set_value(participant[1],'number_of_days_registered',days)

```

Next we write a function to determine the lowest number of both of the columns and write this to a new column called 'days'. This columns will be used to standardize other features.

```

In [23]: for participant in participants_srs.iteritems():
        events_holder = overview.loc[participant[1],['number_of_unique_days','number_of_days_registered']]
        overview.set_value(participant[1],'days',events_holder.min())

```

1.5.10 Patient status Last we add the patient class information column to the overview necessary in order to identify which participants are patients and controls.

1 = Patient 0 = Control

```

In [24]: for participant in participants_srs.iteritems():
        events_holder = behapp_events.loc[idx[participant[1],:],idx['patient']].unique()
        overview.set_value(participant[1],'patient',events_holder[0])

```


1.5.11 Select participants on days Select the participants that conform to the minimal number of days, currently set at 14 days,

```
In [34]: global minimum_days
```

```
overview = overview[overview.days >= minimum_days]
```

1.1.6 1.6 Standardize

1.6.1 Standardize features over days The number of days a participant has been running the monitoring application, and thus the resulting measurements, vary wildly between the participants. In order to make a fair comparison we choose the pragmatic solution of standardizing the measurements according to a common denominator, in this case we calculate the average of each feature per day (column: day).

The column `unique_contacts` is not part of this calculation since it does not represent data that has been measured over time. Last, the column `'places_visited'` will be standardized by a different column called `'places_visited_days_count'` since the GPS data was found not be reliable over all the days where measurements were submitted.

```
In [26]: # First we rename the columns that we intend to use later to a standard format.
```

```
overview.rename(columns={
    "Incoming call": "incoming_calls",
    "Outgoing call": "outgoing_calls",
    "Missed call": "missed_calls",
    "Incoming SMS": "incoming_sms",
    "Outgoing SMS": "outgoing_sms",
    "Twitter personal tweets": "twitter_personal_tweets",
    "Twitter direct incoming": "twitter_direct_incoming",
    "Twitter direct outgoing": "twitter_direct_outgoing",
    "Facebook timeline post": "facebook_timeline_post",
    "Incoming MMS": "incoming_mms",
    "Outgoing MMS": "outgoing_mms",
    "Application activity": "application_activity",
    "Number of detected Bluetooth devices": "bluetooth_detection_events"
}, inplace=True)
```

```
# Determine the columns that will be divided by the 'days' column.
```

```
columns_to_divide = ['incoming_calls',
                     'outgoing_calls',
                     'missed_calls',
                     'call_duration',
                     'incoming_sms',
                     'outgoing_sms',
                     'twitter_personal_tweets',
                     'twitter_direct_incoming',
                     'twitter_direct_outgoing',
                     'facebook_timeline_post',
                     'incoming_mms',
                     'outgoing_mms',
                     'application_activity',
                     'whatsapp_activity',
                     'bluetooth_devices_detected',
                     'Position',
                     'bluetooth_detection_events'
]
```

```

# Perform the standardizing calculation.
for column in columns_to_divide:
    overview[column] = overview[column] / overview['days']

# Standardize the 'places_visited' column with the 'places_visited_days_count'.
overview['places_visited_daily'] = overview['places_visited_daily'] / overview['places_visited']

In [27]: # Selection of columns in the final overview.
overview_std_dly = overview.loc[:,['incoming_calls',
                                   'outgoing_calls',
                                   'missed_calls',
                                   'call_duration',
                                   'incoming_sms',
                                   'outgoing_sms',
                                   'incoming_mms',
                                   'outgoing_mms',
                                   'twitter_personal_tweets',
                                   'twitter_direct_incoming',
                                   'twitter_direct_outgoing',
                                   'facebook_timeline_post',
                                   'application_activity',
                                   'whatsapp_activity',
                                   'bluetooth_devices_detected',
                                   'bluetooth_detection_events',
                                   'bluetooth_device_diversity',
                                   'bluetooth_detection_ratio',
                                   'unique_contacts',
                                   'places_visited_daily',
                                   'places_visited_diversity',
                                   'patient']]

```

1.6.2 Z-Score standardization for analysis To conclude the preparation phase NaN values in the 'places_visited' column (in case of a complete lack of GPS data) are replaced with the value 0. Next we perform the standardscaling method to orient al values around Z-scores.

```

In [28]: # Replace NaN values with 0.
overview_std_dly = overview_std_dly.fillna(value='0')

# Fix incorrect datatypes for the places_visited and patient columns.
#overview['places_visited'] = overview['places_visited'].astype(float)

# Create an array for the columns that will be standardized.
std_columns = overview_std_dly.columns.values
# Drop the patient column, this does not need to be standardized.
std_columns = std_columns[:-1]

# Select the columns that need to be indexed for scaling.
std_scale = preprocessing.StandardScaler().fit(overview_std_dly[std_columns])

# Apply the scaler to the dataset.
overview_scaled = std_scale.transform(overview_std_dly[std_columns])

# Convert the overview to a dataframe.
overview_std_z = pd.DataFrame(overview_scaled)

```

```

# Re-add the person_id's
overview_std_z.insert(0, 'person_id', overview.index)
overview_std_z = overview_std_z.set_index('person_id', drop=True)

# Re-add the column names
i = 0
for column in std_columns:
    overview_std_z.rename(columns={i: column}, inplace=True)
    i = i + 1

# Eskes's score transformation
f = lambda x: ((x+3)/6)*100
overview_eskes = overview_std_z.applymap(f)

# Re-add the patient status
for participant in overview.index:
    events_holder = behapp_events.loc[idx[participant,:], idx['patient']].unique()
    overview_std_z.set_value(participant, 'patient', events_holder[0])
overview_std_z['patient'] = overview['patient'].astype(int)

```

1.2 2. Sociability scoring

1.2.1 2.1 Eskes' (2013) style scoring

In this section we prepare an overview which depicts the 'sociability' score of each participant. The scoring practice is based on the sociability score model as defined by Eskes (2013). The scores are composed of the average of two subscores, the 'social exploration' and the 'communication' score. We start by including the features as originally devised by Paul Eskes:

- Communication score
 - incoming_calls
 - outgoing_calls
 - call_duration
- Social exploration score
 - places_visited_daily
 - bluetooth_devices_detected

```

In [29]: # Original Eskes (2013) style sociability scoring
eskес_original_communication_score = pd.DataFrame(overview_eskes[["incoming_calls",
                                                                    "outgoing_calls",
                                                                    "call_duration"]].mean(axis=

eskес_original_communication_score.rename(columns={0: 'communication_score'}, inplace=True)

eskес_original_social_exploration_score = pd.DataFrame(overview_eskes[["places_visited_daily",
                                                                    "bluetooth_devices_dete

eskес_original_social_exploration_score.rename(columns={0: 'social_exploration_score'}, inplace=True)

eskес_original_score = eskес_original_communication_score.join(eskес_original_social_exploration_score)

# Re-add the patient status
for participant in overview.index:
    events_holder = behapp_events.loc[idx[participant,:], idx['patient']].unique()

```

```

        eskes_original_score.set_value(participant,'patient',events_holder[0])
eskes_original_score['patient'] = eskes_original_score['patient'].astype(int)

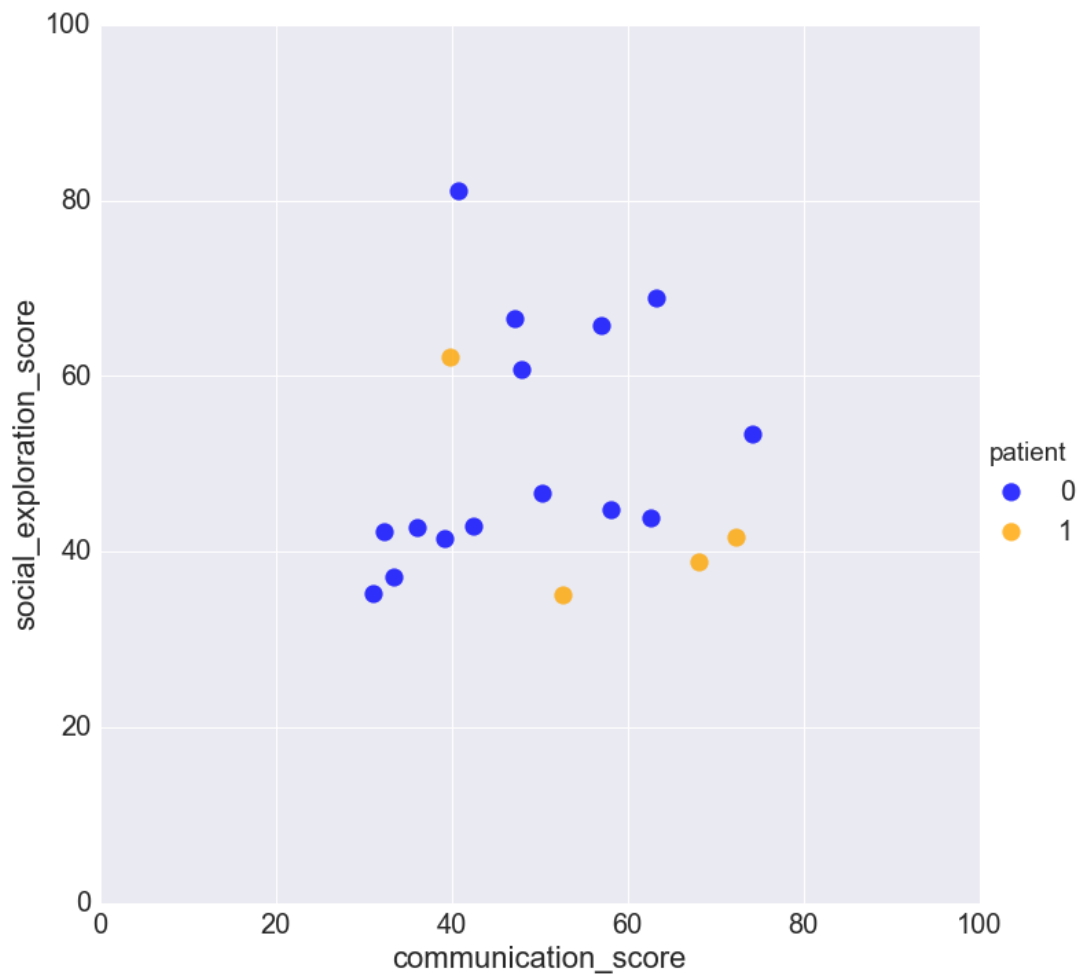
sns.set(font_scale=2)
eskes_plot = sns.lmplot(palette=['blue','orange'],
                        x="communication_score",
                        y="social_exploration_score",
                        hue="patient",
                        data=eskes_original_score,
                        size=10,
                        fit_reg=False,
                        scatter_kws={"s": 175},
                        markers=["o", "o"])

eskes_plot.set(ylim=(30,90))
eskes_plot.set(xlim=(30,90))

/Users/raj/anaconda/lib/python2.7/site-packages/matplotlib/collections.py:590: FutureWarning: elementwise
if self._edgecolors == str('face'):

Out[29]: <seaborn.axisgrid.FacetGrid at 0x10977f850>

```



1.2.2 2.1 Modernized sociability scoring

Next we adjust the subscores by including additional features in the communication and the social exploration scores.

- Communication score
 - incoming_calls
 - outgoing_calls
 - missed_calls
 - call_duration
 - unique_contacts
 - whatsapp_activity
- Social exploration score
 - places_visited_daily
 - bluetooth_devices_detected
 - bluetooth_device_diversity

In [33]: *# Modernized sociability scoring*

```
modernized_communication_score = pd.DataFrame(overview_eskes[["incoming_calls",
                                                             "outgoing_calls",
                                                             "missed_calls",
                                                             "call_duration",
                                                             "unique_contacts",
                                                             "whatsapp_activity",
                                                             ]].mean(axis=1))

modernized_communication_score.rename(columns={0: 'communication_score'}, inplace=True)

modernized_social_exploration_score = pd.DataFrame(overview_eskes[["places_visited_daily",
                                                                    "bluetooth_devices_detected",
                                                                    "bluetooth_device_diversity",
                                                                    ]].mean(axis=1))

modernized_social_exploration_score.rename(columns={0: 'social_exploration_score'}, inplace=True)

eskes_modernized_score = modernized_communication_score.join(modernized_social_exploration_score)

# Re-add the patient status
for participant in overview.index:
    events_holder = behapp_events.loc[idx[participant,:],idx['patient']].unique()
    eskes_modernized_score.set_value(participant, 'patient', events_holder[0])
eskes_modernized_score['patient'] = eskes_original_score['patient'].astype(int)

eskes_modernized_plot = sns.lmplot(palette=['blue', 'orange'],
                                  x="communication_score",
                                  y="social_exploration_score",
                                  hue="patient",
                                  data=eskes_modernized_score,
                                  size=10,
                                  fit_reg=False,
                                  scatter_kws={"s": 175},
```

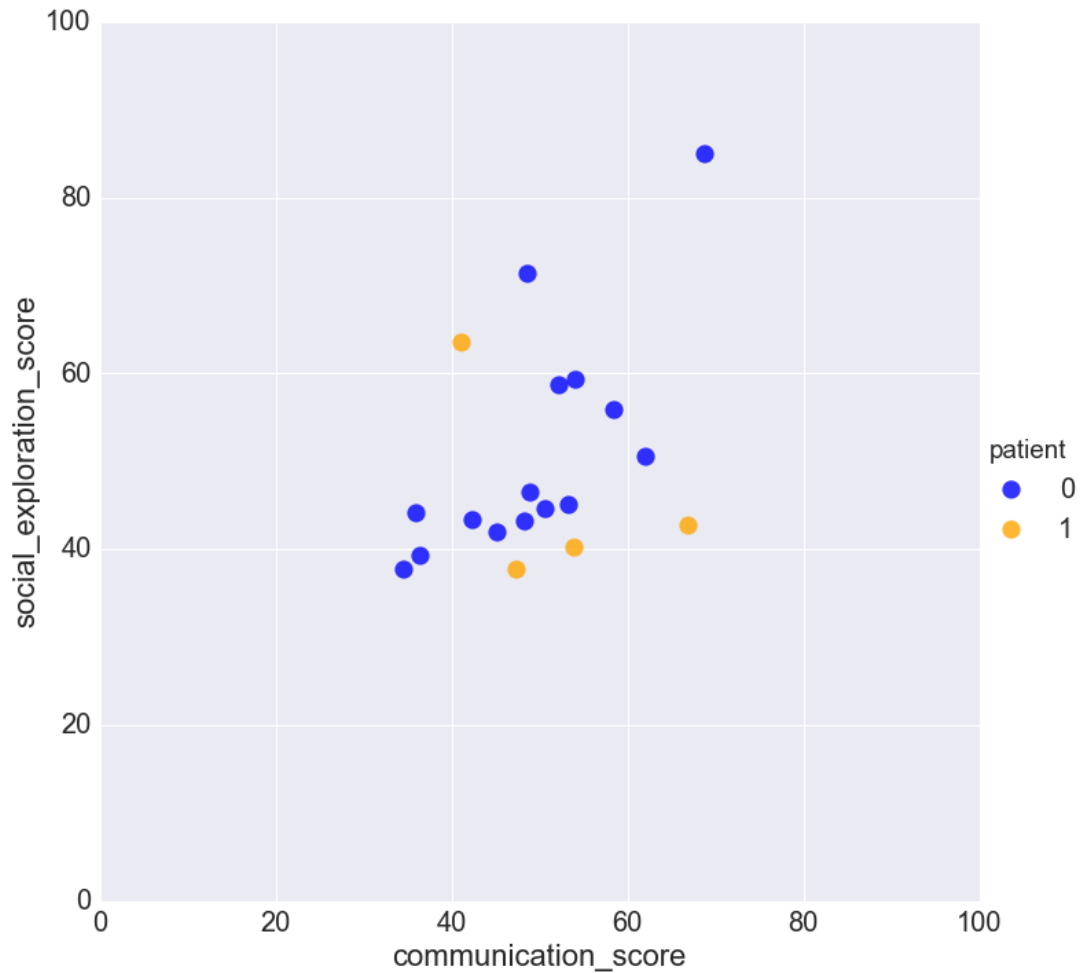
```

markers=["o", "o"])

eskes_modernized_plot.set(ylim=(0,100))
eskes_modernized_plot.set(xlim=(0,100))

Out[33]: <seaborn.axisgrid.FacetGrid at 0x108fc8110>

```



```

In [31]: print eskes_original_score
         print eskes_modernized_score

```

communication_score	social_exploration_score	patient
person_id		
13	63.360748	68.900714
14	40.890315	81.039556
34	36.182303	42.662368
78	32.403567	42.175195
80	74.216944	53.269569
82	33.465820	37.038225
98	31.166105	35.177499

99	58.117945	44.680684	0
100	47.166051	66.527582	0
103	72.372813	41.642544	1
104	42.545025	42.855448	0
105	39.895389	62.180132	1
106	57.042015	65.764014	0
107	39.194943	41.391505	0
111	52.763081	34.951846	1
114	68.212922	38.820343	1
116	50.399389	46.538623	0
117	62.648041	43.718548	0
118	47.956584	60.665605	0
	communication_score	social_exploration_score	patient
person_id			
13	68.862679	84.974619	0
14	48.688146	71.335499	0
34	42.445929	43.339840	0
78	36.038773	44.121686	0
80	62.030455	50.563751	0
82	36.525847	39.223418	0
98	34.655974	37.666754	0
99	50.647362	44.555525	0
100	52.208295	58.600686	0
103	66.932637	42.665603	1
104	48.379285	43.174965	0
105	41.121884	63.561700	1
106	54.090725	59.339420	0
107	45.152680	41.899762	0
111	47.431293	37.651825	1
114	54.012612	40.129193	1
116	49.010547	46.381341	0
117	53.332663	45.026374	0
118	58.432214	55.788037	0

Appendix F

ICIS 2016 Paper submission

A Recipe for Machine Learning: Meta-algorithmic Modeling for Transparency

Journal:	<i>International Conference on Information Systems 2016</i>
Manuscript ID	ICIS-0697-2016
Track:	03. Data Science and Business Analytics
Keywords:	Algorithm evaluation and selection, Data analysis, Design Science, Machine learning, Method, 03. Data Science and Business Analytics
Abstract:	<p>Despite the growing usage and popularity of machine learning techniques in data mining projects, correctly applying these techniques remains a challenge. We identify the following three main challenges: depth versus breadth, selection versus configuration, and accuracy versus transparency. To counter these challenges, we propose a meta-algorithmic modelling approach to reuse state-of-the-art ML knowledge and best practices in the appropriate application of ML techniques, whilst at the same time provide information on how to cope with challenges like parameter optimization and model transparency.</p> <p>This research-in-progress aims to provide highly understandable and deterministic methodological recipes to guide researchers without in-depth ML expertise step-by-step through an optimized ML process, working towards one well-defined and transparent methodological infrastructure for applied data science.</p>

A Recipe for Machine Learning:

Meta-algorithmic modeling for transparency

Research-in-Progress

Introduction

With the steadily growing availability of data storage space and computing power, advanced data mining efforts are coming in reach of increasingly more people. One way to perform a data mining project, and central to this research-in-progress, is the application of machine learning (ML) techniques. The application of ML techniques spans various disciplines like mathematics, statistics and computer science. These disciplines combined support the act of learning and result in models that are fitted to data. The challenge is to derive models that are accurate in the sense that they reflect the underlying patterns in the data whilst ignoring peculiarities that do not represent reality. A popular and well known purpose of these models is to make predictions on new (and unseen) examples of data. However, ML techniques are also well suited to explore the underlying patterns of a dataset. More often than not, machine learning techniques are employed to learn about the structure of a data set (Hall et al. 2011).

Problem Statement

Despite the growing usage and popularity of machine learning techniques in data mining projects, correctly applying these techniques remains a challenge. We list the three main challenges below:

1. *Depth versus breadth:* The ML field knows many different use cases, each of which has a sizeable body of literature surrounding the specific cases. The literature is usually found to be heavy on mathematical terminology and aimed at the computer science community. This prevents researchers from other fields in learning and (correctly) applying machine learning techniques in their own research (Domingos 2012).
2. *Selection versus configuration:* In line with the aforementioned, applying machine learning techniques confronts users with many degrees of freedom in how to assemble and configure a learning system. One example of this is the fact that algorithm performance is largely determined by parameter settings, these settings are specific for each class of algorithm. However, in practice end users usually do not have enough knowledge on how to find optimal parameter settings (Yoo et al. 2012). Many users leave the parameters to their default settings and base algorithm selection on reputation and / or intuitive appeal (Thornton et al. 2013). This may lead to researchers using underperforming algorithms and gaining suboptimal results.
3. *Accuracy versus transparency:* Concerning the creation of models: ML shows that currently there is a trade-off to be had between accuracy and transparency (Kamwa et al. 2012). In practice this means that algorithms which yield a high amount of insight into the data do not perform as well as their non-transparent (black box) counterparts and the other way around.

We follow a meta-algorithmic modelling approach to reuse state-of-the-art ML knowledge and best practices in the appropriate application of ML techniques, whilst at the same time provide information on how to cope with challenges like parameter optimization and model transparency (Pachidi et al. 2014). Our goal is to provide highly understandable and deterministic methodological *recipes* to guide researchers without in-depth ML expertise step-by-step through an optimized ML process (Vleugel et al. 2010), based on the design science research approach (Hevner et al. 2004).

Research Approach

By taking into account our problem statement the overarching research question of this research-in-progress is formulated as follows:

How can a domain independent method be developed to guide the process of constructing transparent machine learning models?

We will initially proceed with a limited scope: the creation of method fragments focused on supervised machine learning for binary classification tasks on structured data. This type of machine learning is concerned with deriving models from (training) data that are already available. Coincidentally this is one of the most applied and mature areas within the machine learning practice (Kotsiantis et al. 2007).

First a theoretical foundation is established on the subjects of data mining, machine learning and model transparency. The concepts derived from this foundation are then grouped using the structure of a data mining process model. For our purposes we apply the base structure of the CRISP-DM process model and group the concepts into the following phases: data understanding, data preparation, and modeling & evaluation. Our method fragments will be composed using the same structure. In order to design the method fragments we will turn to the practice of method engineering (ME). Method engineering is defined as "the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems" (Brinkkemper 1996). From this discipline we will apply the meta-modelling approach (Weerd et al. 2008), this yields a process-deliverable diagram (PDD). A PDD consists of two diagrams, the left-hand side shows an UML activity diagram (processes) and the right-hand side shows an UML class diagram (concepts / deliverables). Both diagrams are integrated and display how the activities are tied to each deliverable. Lastly, the activities and the concepts are each explained in separate tables. Due to page restrictions these explanatory tables are excluded from this paper.

On Model Transparency

The concept of model transparency occasionally surfaces in the body of literature. In particular, when it concerns decision support systems where it must be clear how a system came to a certain (classification) decision (Johansson et al. 2004; Olson et al. 2012; Kamwa et al. 2012b; Allahyari et al. 2011).

There is consensus in the literature about the types of algorithms that are known to yield transparent and non-transparent (black box) models. Both tree and rule models are considered as transparent and highly interpretable. On the other hand, artificial neural networks, support vector machines and ensembles like random forests are considered as black boxes (Johansson et al. 2004; Olson et al. 2012; Kamwa et al. 2012b).

Currently there is no common ground on the subject of tree and rule model complexity. Although considered as transparent, critics note that the interpretative value of complex tree and rule models should be questioned (Johansson et al. 2004). On the other hand, a study on model understandability found indications that the assumption where simpler models are considered as more understandable does not always hold as true (Allahyari et al. 2011).

The choice between a transparent and non-transparent modeling technique is not immediately obvious since there is a tradeoff to be had between accuracy and transparency. Black box modeling techniques have better classification / prediction performance and that the tradeoff with better interpretable solutions is unavoidable. We found two solutions in the body of literature that aim to bridge this gap.

The first solution is aimed towards extracting comprehensible information in the form of rules and trees from black box modeling techniques like artificial neural networks and support vector machines (Johansson et al. 2004; Martens et al. 2007; Setiono 2003). The practice delivers comprehensible information but is criticized for being unrepresentative of the original model due to oversimplification (Cortez et al. 2013).

The second solution approaches the problem from the opposite direction by improving the performance of a transparent modeling technique to a level where it competes with its black box counterparts. A variant of linear modeling is applied known as generalized additive modeling (GAM) enriched with information on pairwise interactions between features (Lou et al. 2013). This allows to retain the explanatory value of linear models and at the same time achieve high performance in terms of classification accuracy. The technique exposes the contribution of each feature in relation to the outcome values.

Method Fragments

In this section we present the method fragments as derived from our literature study on the domains of data mining and machine learning. All fragments are accompanied with a text description.

Data understanding

Before starting with any data mining project it is important to become familiar with the data that will be analyzed. The goal is to improve one's understanding of the data by using (statistical) tools to summarize, plot and review datapoints in the data set. This practice is called exploratory data analysis (EDA) (Tukey 1977).

The data understanding phase as depicted in Figure 1 revolves around the application of exploratory data analysis (EDA) techniques to generate visualizations and tables to gain a first insight into the relationships between the features of a data set. A high number of features can make these deliverables difficult to interpret. Therefore, the activity flow shows that in cases of high dimensional data sets it is recommended to pre select a subset of features using a feature selection technique.

We recommend the creation of histogram graphs, pairwise scatterplots and correlation matrices to start exploring relationships between the features of a dataset. Histogram graphs and pairwise scatterplots serve the purpose of visualizing overlap / separability between the various classes of a data set. Feature correlation matrices are used to determine which features are redundant, these should be removed when applying the naive bayes (probabilistic) model.

Data preparation

The data preparation phase (Figure 2) consists of three main activities. The data set construction activity entails loading the raw data and engineering new features based on the raw data. Feature engineering can be a substantial task but is difficult to capture in a method since it is highly situational. The last task within this activity is feature selection. Not all features in a given data set have the same informative importance or any importance at all. This can be problematic as some classification algorithms are designed to make the most of the data that is presented to them. In these cases even irrelevant features will eventually be included in the model. In other words the model will be overfitted to the data which means that the classification algorithm has included the noise as an integral part of the model (Tang, Alelyani, and Liu, 2014). The solution is to select a subset of only the most informative features reducing the dimensionality (number of features) of the data set in the process. Feature selection is either performed manually using EDA techniques, or selection is performed using a feature selection algorithm.

The feature extraction activity entails the application of projection methods. Projection methods like principal component analysis are automated feature engineering techniques that aim to best describe the main differentiators of a data set creating a select (low) number of features in the process (dimensionality reduction). Transparency between the outcome variable and the original features may be lost while using a projection technique.

Lastly, the modeling technique preparation activity consists of three paths that define preparation steps depending on the model type chosen by the data scientist. When tree and rule models are required due to model transparency concerns, no additional preparation steps are necessary since modern algorithm implementations take care of preparation steps internally. Linear models and the probabilistic naive Bayes model can be chosen due to performance concerns. Both types require their own conversion steps in order to be able to process the data in the next phase of the DM process. The naive Bayes model type e.g. requires redundant features to be removed since they will negatively influence classifier results. Linear model types require input data to be represented in numerical form so transformation steps should be performed as needed e.g. the binarization of categorical data. Note however that some concrete algorithm implementations of linear models may perform these steps as part of their internal workings.

Modeling & Evaluation

The modeling and evaluation method fragment (Figure 3) consists of three activities aimed at deriving classification models from data sets. The search space definition activity has a route to explore fully

automated model (and parameter) selection in analyzing the data set. Currently one experimental implementation exists in the form of Auto-WEKA (Thornton et al. 2013). Auto-WEKA is an experimental machine learning toolkit that almost completely relies on Bayesian optimization techniques to generate models. The toolkit is unique in the sense that it considers the choice for the modeling technique as part of the problem space as well. This relieves potential users from having to manually select and test algorithms, instead Auto-WEKA uses all the algorithms that are part of the WEKA toolkit and determines which algorithm generates the best results for a given data set. Currently, due to the novelty of this technique, the approach should be used to gain initial insight into model types that may perform best on the provided data set.

Next the application of automated search strategies is central to the following activity named “find optimal parameters”. Recall from our introduction that the performance of algorithms is dependent on how they are configured, a problem known as (hyper) parameter optimization. Getting optimal performance from a modeling technique means finding the right (combination of) parameter settings. The best settings will be different for each data set which necessitates an automated means of determining these values. Search strategies like grid search, random search and Bayesian optimization support the task to (intelligently) iterate over combinations of parameters evaluating the performance at each attempt. This task requires the data scientist to decide on various factors that determine how the search will be executed. The following factors should be considered:

- The first factor comprises the parameters that belong to a specific model type. Parameter types can range from procedural configuration settings to the specific number of times a procedure is performed.
- The model type itself. The data scientist can choose to iterate over different model types as well (tree, rule, ensemble, linear and probabilistic) to find out which type works best given a specific data set. This approach is similar to Auto-WEKA since it also includes the model type as part of the problem (search) space.
- The performance measure(s) used to evaluate each attempt. Common measures are classification accuracy, true positive rate (TPR), false positive rate (FPR) and the area under the curve (AUC). Using a combination of measures is necessary since classification accuracy by itself is known to misrepresent the performance of a model in the case of class imbalances in the data set.
- The resampling method used to support the evaluation process. Resampling methods apply various procedures to train and test models on the data provided to them. For example, the holdout method splits the data set in a training and test set, usually in a 70% - 30% ratio. The model is first trained using the training set, afterwards it is tested on the unseen instances of the test set. Other resampling methods include: (stratified) k-fold cross validation, leave-one-out and bootstrapping. The search strategy itself. Grid search is exhaustive by nature meaning that all possible parameter combinations will be tried. This can be costly both in time and computing resources. Random search and Bayesian optimization aim to find the optimal set of parameters intelligently requiring significantly less tries to do so.

The factors discussed above are common to the search strategies outlined in this section, combined they form the template that makes up the complete problem space through where the search will be executed. The structure and accessibility of this approach is in line with the design goal of this research project where we aim to construct a method that enables a user to create optimal models.

Lastly, the activity “predict & classify” is followed to conclude a DM project. The model derived from the parameter search activity can now be used to classify new and unseen data.

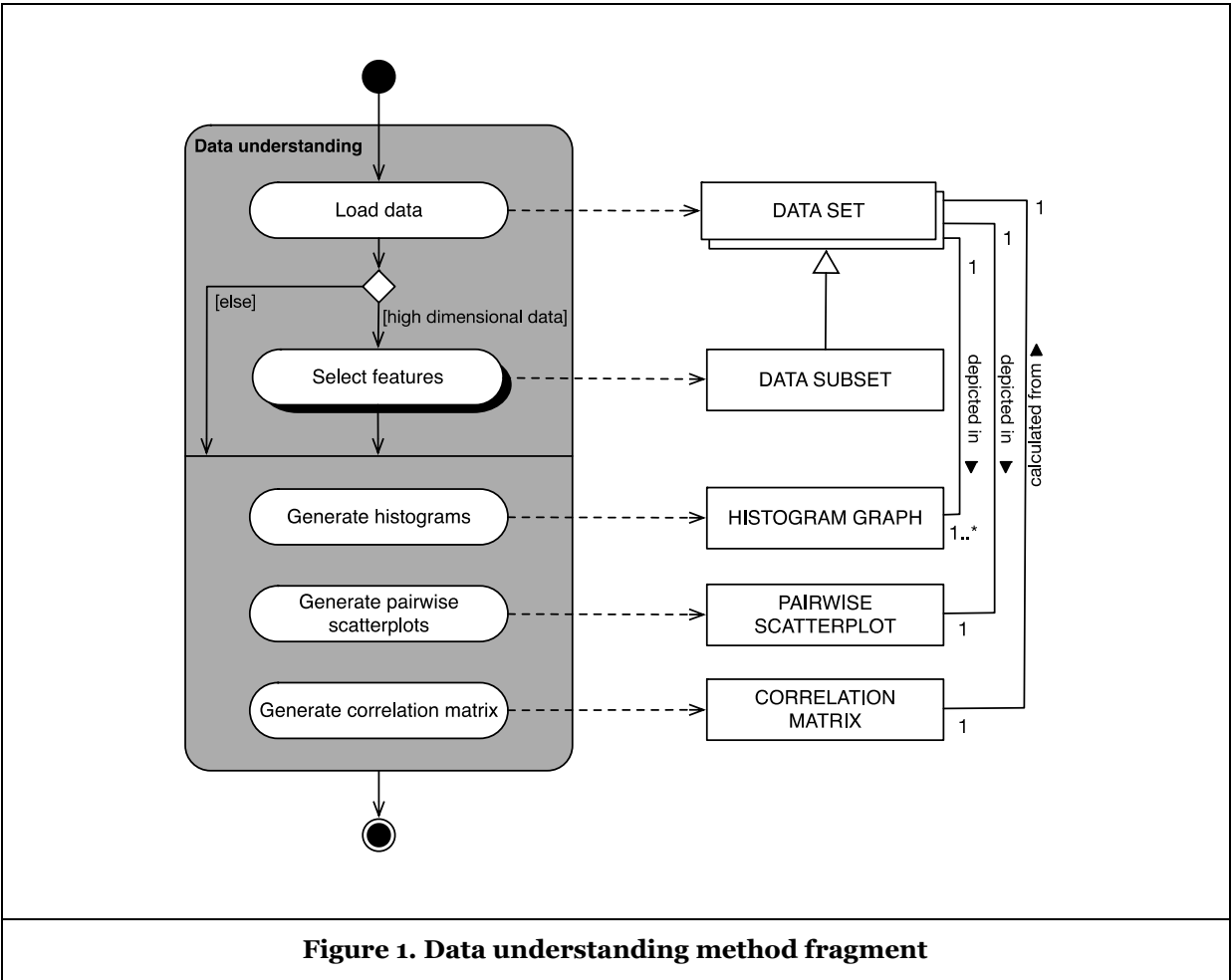
Future research

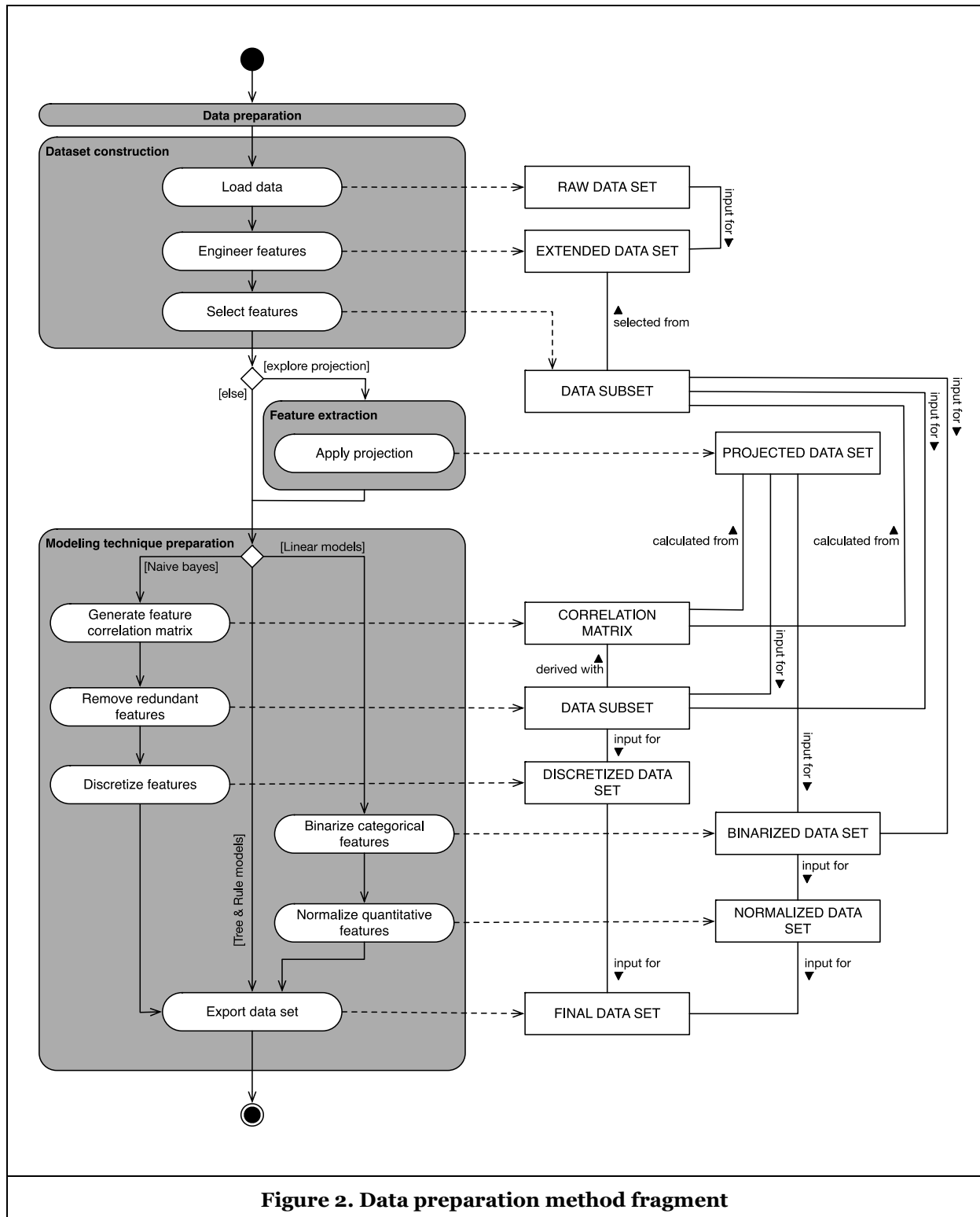
We are currently extending and refining the method fragments as outlined in Figures 1 to 3 with the goal to ultimately evaluate the method on a broad array of data sets: ranging from small/large to low/high dimensional data sets. We are curious to see how classification performance holds up over different variants in data sets. We are also interested, by using qualitative research methods, in studying to what extent the methods support non data scientists in their efforts to perform DM projects.

Next, the problem space of our research could be broadened to cover cases outside of the domain of supervised binary classification, e.g. multiclass, regression and image analysis problems. Method fragments could be created to deal with (sub)cases in the aforementioned domains.

Furthermore, the structures defined in these methods could be used for the development / enhancement of data mining tools. Auto-WEKA is an example of such a tool but follows a rigid method. For example, the tool uses a pre set path of actions and tasks and does not support embedding domain knowledge during the DM process. From our own experience we identify a great need for sophisticated tools that offer simplified access to advanced ML techniques while retaining the ability to embed domain knowledge in the data mining process.

Finally, we aim to further refine and integrate existing meta-algorithmic models, as well as to incrementally yet continuously broaden our modeling scope in creating ML method fragments to also include unsupervised learning, non-binary classification tasks, and unstructured data, among others. As our strategic objective we envision one well-defined, transparant methodological infrastructure for applied data science which interconnects the vast body of knowledge as recipes for machine learning.





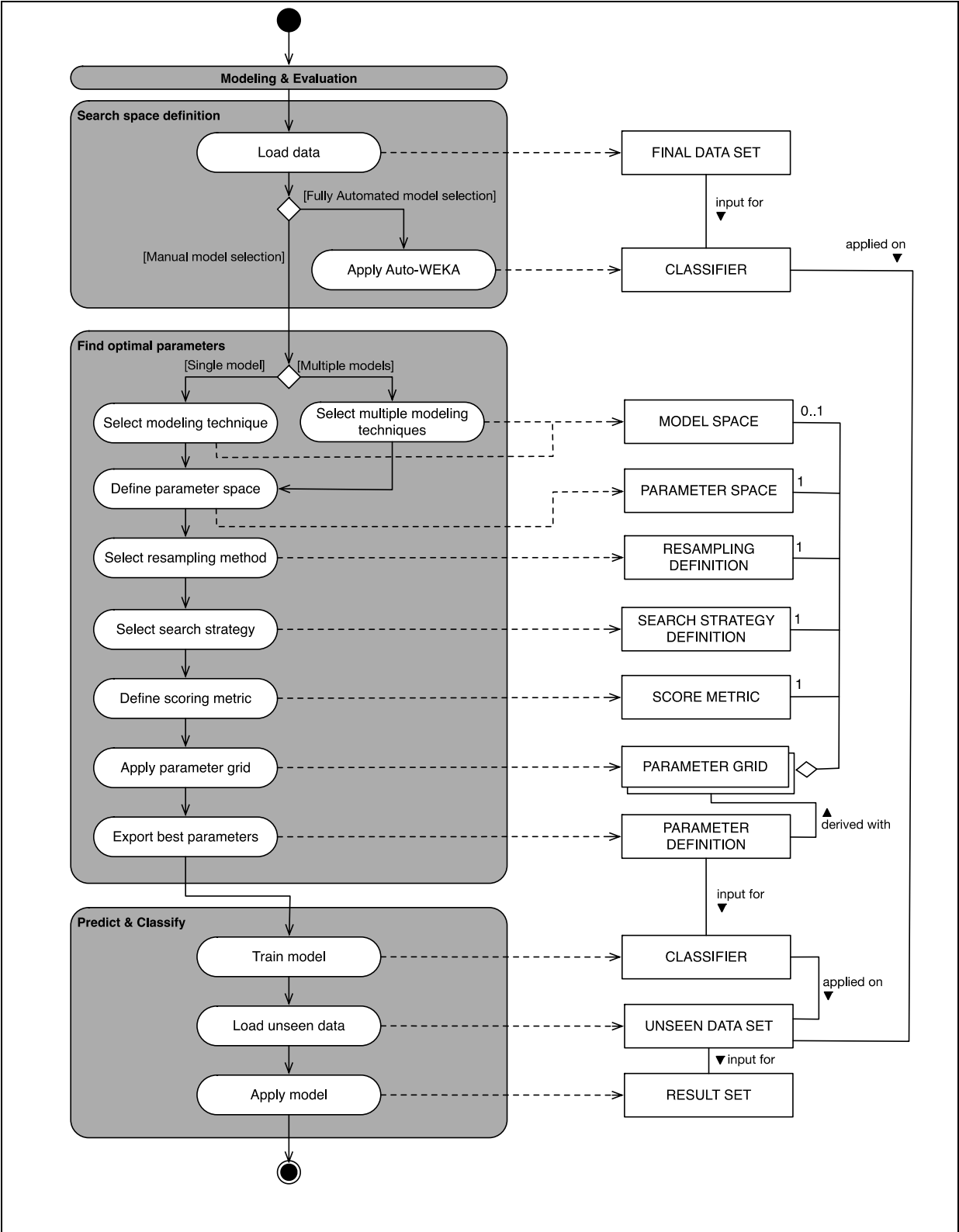


Figure 3. Modeling & evaluation method fragment

References

- Allahyari, H., and N. Lavesson. 2011. "User-Oriented Assessment of Classification Model Understandability," in *11th Scandinavian Conference on Artificial Intelligence*, pp. 11-19.
- Brinkkemper, S. 1996. "Method Engineering: Engineering of Information Systems Development Methods and Tools," *Information and Software Technology* (38:4), pp. 275-280.
- Cortez, P., and M. J. Embrechts. 2013. "Using Sensitivity Analysis and Visualization Techniques to Open Black Box Data Mining Models," *Information Sciences* (225), pp. 1-17.
- Domingos, P. 2012. "A Few Useful Things to Know about Machine Learning," *Communications of the ACM* (55:10), pp. 78-87.
- Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. 2009. "The WEKA Data Mining Software: An Update," *ACM SIGKDD Explorations Newsletter* (11:1), pp. 10-18.
- Hevner, A., S. March, P. Jinsoo, and S. Ram. 2004. "Design Science in Information Systems Research," *MIS Quarterly* (28:1), pp. 75-105.
- Johansson, U., L. Niklasson, and R. König. 2004. "Accuracy Vs. Comprehensibility in Data Mining Models," in *Proceedings of the Seventh International Conference on Information Fusion Vol. 1*, pp. 295-300.
- Kamwa, I., S. Samantaray, and G. Joós. 2012. "On the Accuracy Versus Transparency Trade-Off of Data-Mining Models for Fast-Response PMU-Based Catastrophe Predictors," *IEEE Transactions on Smart Grid* (3:1), pp. 152-161.
- Kotsiantis, S. B., I. Zaharakis, and P. Pintelas. 2007. "Supervised Machine Learning: A Review of Classification Techniques," in *Emerging Artificial Intelligence Applications in Computer Engineering*, pp. 3-24.
- Lou, Y., R. Caruana, J. Gehrke, and G. Hooker. 2013. "Accurate Intelligible Models with Pairwise Interactions," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 623-631.
- Olson, D. L., D. Delen, and Y. Meng. 2012. "Comparative Analysis of Data Mining Methods for Bankruptcy Prediction," *Decision Support Systems* (52:2), pp. 464-473.
- Pachidi, S., M. Spruit, and I. van der Weerd. 2014. "Understanding Users' Behavior with Software Operation Data Mining," *Computers in Human Behavior* (30), pp. 583-594.
- Setiono, R. 2003. "Techniques for Extracting Classification and Regression Rules from Artificial Neural Networks," *Computational Intelligence: The Experts Speak Piscataway, NJ, USA: IEEE*, pp. 99-114.
- Tang, J., S. Alelyani, and H. Liu. 2014. "Feature Selection for Classification: A Review," *Data Classification: Algorithms and Applications Vol. 37*, pp. 2 – 29.
- Thornton, C., F. Hutter, H. H. Hoos, and K. Leyton-Brown. 2013. "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 847-855.
- Tukey, J. W. 1977. *Exploratory Data Analysis*, Addison-Wesley.
- van de Weerd, I., and S. Brinkkemper. 2008. "Meta-Modeling for Situational Analysis and Design Methods," *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications*, pp. 35-54.
- Vleugel, A., M. Spruit, and A. van Daal. 2010. "Historical data analysis through data mining from an outsourcing perspective: the three-phases method," *International Journal of Business Intelligence Research*, (1:3), pp. 42-65.
- Yoo, I., P. Alafaireet, M. Marinov, K. Pena-Hernandez, R. Gopidi, J. Chang, and L. Hua. 2012. "Data Mining in Healthcare and Biomedicine: A Survey of the Literature," *Journal of Medical Systems* (36:4), pp. 2431-2448.